U.S. DEPARTMENT OF COMMERCE
National Technical Information Service
PB-284 155

A Primer for MLAB (Mathematical Modeling Laboratory) An On-Line Modeling Laboratory

ADP/First Data, Inc, Washington, D C Federal Systems Div

Prepared for

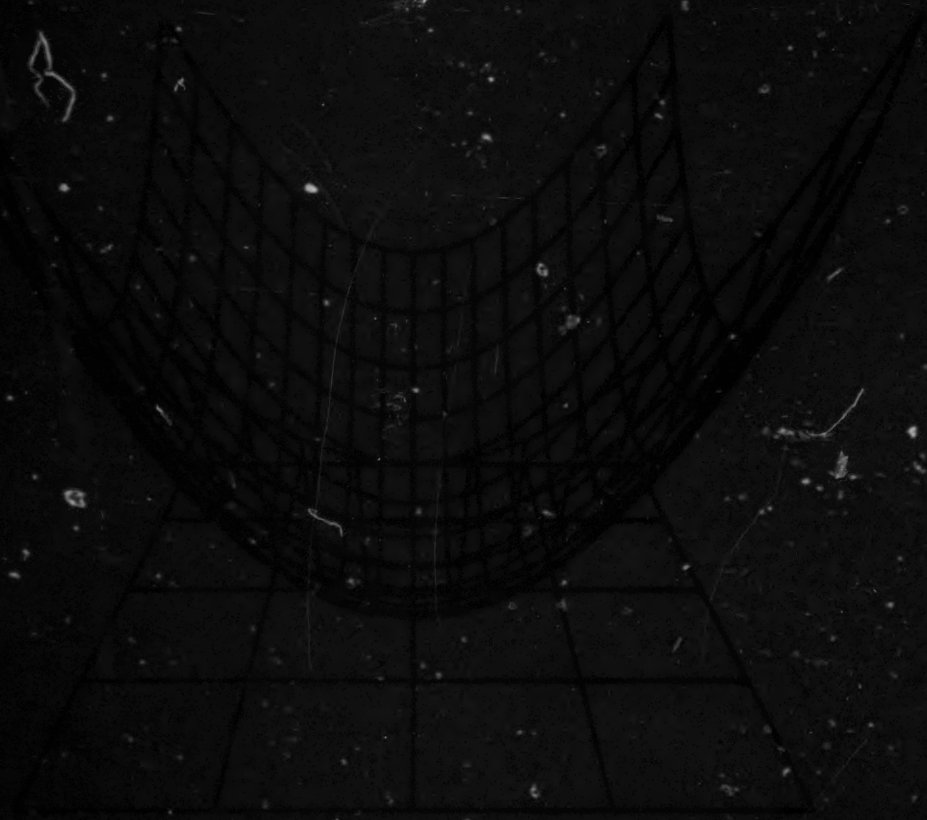
Environmental Protection Agency, Washington, D C Management Information and Data Systems Div

Apr 78

PB 284 155

A Primer on CSMILAB

An On-Line Modeling Laboratory



REPRODUCED BY
NATIONAL TECHNICAL
INFORMATION SERVICE
U. S. DEPARTMENT OF COMMERCE
SPRINGFIELD, VA. 22161

MLAB PRIMER

APRIL, 1978

PREPARED BY

DR. EMILE TOBENFELD, DR. C. JOHN MC CANN, AND ROBERT M. YOUNG

FOR

ENVIRONMENTAL PROTECTION AGENCY
MANAGEMENT INFORMATION AND DATA SYSTEMS DIVISION
401 M STREET, S.W.
WASHINGTON, D.C. 20460

INTRODUCTION

The Mathematical Modeling Laboratory (MLAB) described in this PRIMER was developed by Gary Knott and Douglas Reece of the National Institutes of Health in Bethesda, Maryland. It was based, in part, on the earlier work done by Richard Shrager at N.I.H. and uses the curve fitting routine which he developed. It runs on a DEC SYSTEM 10 or 20 timesharing system and is available for public distribution.

MLAB is available to EPA personnel through First Data Corporation (FDC), a subsidiary of Automatic Data Processing, Inc., under contract with EPA to provide mathematical modeling support and services, of which MLAB is an integral part. This PRIMER was developed by First Data Corporation.

MLAB was developed as a tool for experimentation with and evaluation of mathematical models or functions. Its power and flexibility allow it to be used cost effectively for a wide spectrum of mathematical problems ranging from elementary arithmetic operations through matrix operations, systems of linear and nonlinear equations, differental equations, and, in particular, curve fitting and graphics.

The true power of MLAB resides in its simplicity and in the fact that it is used interactively by the researcher. Many different models or different variants of a single model can be tried out in a single session since the computer response is immediate. Sophisticated operations can be accomplished on-line by issuing a few relatively simple commands and since results are available immediately to the user, he can update his model and proceed to the next operation based on the results he has received.

MLAB has been written in such a way that the user need not be familiar with the operation of a computer nor need the user be conversant in any programming language. By following the procedures outlined in this PRIMER the user should be able to take advantage of much of the power and flexibility of MLAB.

More detailed instructions and information about MLAB are contained in the MLAB USER'S MANUAL.

HOW TO USE THE PRIMER

The PRIMER has been written so that the reader may be prepared to use MLAB almost immediately. After the first few pages of Section 2 in Chapter 1, the reader should be able to perform elementary hand calculator type problems using MLAB. By the end of Chapter 1, the reader should be able to enter data into matrices and to draw curves through data points. Many examples are interspersed throughout the text and, since the examples labeled EXAMPLE MLAB J are complete MLAB sessions, the reader can use these examples as prototypes for any examples he wishes to try out.

Chapters 2 and 3 deal with curve fitting and differential equations and proceed from elementary examples to the more complex. The reader is encouraged to try out a few of the more elementary examples on MLAB and then proceed to the more complex after having had at least a routine brush with the FIT and INTEGRATE operators.

Chapter 4 introduces a few of the special commands in MLAB: the ROOT operator, the SUM command and the FOR command. These are basic commands and do not depend on anything in Chapters 2 and 3. APPENDIX A contains other basic commands and the reader is invited to browse through APPENDIX A and to try out the commands described there.

Chapter 5 contains a discussion of files and in particular a description of DO FILES. DO FILES are the mechanism within MLAB to develop a programming language; that is, a sequence of executable commands which can be called in as a package to operate on data.

Finally, Chapter 6 is essentially comprised of detailed examples illustrating the contents of the first five chapters. A few new, but elementary, commands have been introduced in this chapter. The main thrust of Chapter 6 is to illustrate the preceding material and to indicate some advanced directions that the user of MLAB may pursue.

LOGGING ON

Prior to using the First Data timesharing system in general, and MLAB in particular, the user must be assigned an account number, a user number, and a password. An EPA user must request an account number through his or her local ADP coordinator. Refer to Section 1 of Chapter 1 for a complete listing of the steps involved in logging on.

Concerning any questions which may arise about MLAB, EPA personnel may contact the EPA coordinator for MLAB

MR. DAVID M. CLINE
PROJECT OFFICER
ENVIRONMENTAL RESEARCH LAB
COLLEGE STATION ROAD
ATHENS, GEORGIA 30605
(404) 546-3123

or the First Data coordinator for MLAB

MR. ROBERT M. YOUNG
FEDERAL SYSTEMS DIVISION
ADP FIRST DATA, INC.
2011 EYE STREET, NW
WASHINGTON, D.C. 20006
(202) 872-0580

TABLE OF CONTENTS

CHAPTER I -- BASIC COMMANDS.....1

SECTION 1 - LOGGING ON.....2
SECTION 2 - ARITHMETIC IN MLAB.....4
SECTION 3 - ELEMENTARY MATRIX OPERATIONS.....7
SECTION 4 - THE POINTS OPERATOR AND THE DRAW COMMAND..15
SECTION 5 - THE TTYDRAW COMMAND.....22
SECTION 6 - ERRORS, CORRECTIONS AND ERROR MESSAGES....26

CHAPTER II -- ELEMENTARY CURVE FITTING.....31

CHAPTER III -- DIFFERENTIAL EQUATIONS.....41

SECTION 1 - THE DIFF AND INTEGRAL OPERATORS.....41
SECTION 2 - FIRST ORDER DIFFERENTIAL EQUATIONS.....46
SECTION 3 - SECOND ORDER DIFFERENTIAL EQUATIONS.....50

CHAPTER IV -- SOME SPECIAL MLAB COMMANDS.....55

SECTION 1 - FUNCTION DEFINITIONS.....55
SECTION 2 - ROOTS OF EQUATIONS.....59
SECTION 3 - THE FOR COMMAND AND THE SUM COMMAND.....62

CHAPTER V -- FILES AND THE TEXT EDITOR.....66

SECTION 1 - THE SAVE AND USE COMMANDS.....66
SECTION 2 - THE TEXT EDITOR.....69
SECTION 3 - DO FILES.....72

CHAPTER VI -- SOME DETAILED EXAMPLES.....76

APPENDICES

INDEX

CHAPTER 1: BASIC COMMANDS

MLAB is an interactive mathematical modeling and graphics language which can perform the following explicitly:

1. ARITHMETIC OF SCALARS
2. ALGEBRA OF MATRICES
3. ALGEBRAIC AND ANALYTIC OPERATIONS ON FUNCTIONS
4. CURVE FITTING
5. DRAW CURVES
6. SOLVE DIFFERENTIAL EQUATIONS
7. ELEMENTARY STATISTICS
8. BE USED AS A PROGRAMMING LANGUAGE.

In conjunction with the operators and functions built-in to MLAB, number 1 above simply means that MLAB can be used to perform all the operations that can be performed on a hand calculator. Under number 2 above are included inverses, transposes, determinants, eigenvalues and eigenvectors as well as multiplication and addition of matrices.

Included in number 3 are addition, multiplication, division, and composition of functions and derivatives and integrals of functions. Number 4 refers to the MLAB curve fitting routine (a Marquardt-Levenberg least squares routine) which can be used to fit any specified type of function to given data.

MLAB can draw the graphs of functions; draw smooth curves through data points; label points, curves, axes and graphs. In particular, MLAB can be used to compare the graph of a fitting function to a curve through the original data.

MLAB can also be used as a programming language by constructing files of MLAB executable statements (DO FILES).

As indicated above MLAB can be used to solve first order differential equations or systems of first order equations. Elementary statistics such as means, standard deviations, covariances and correlation coefficients can be computed using MLAB.

I.1 LOGGING ON

To enter MLAB the following steps are taken:

(1) dial the local FDC number;

(2) attach the phone to the acoustic coupler;

(3) press a control P (that is, depress the CTRL key and at the same time depress the P key);

(4) in response to the request for the account-user number, type in these numbers and then press a carriage return;

(5) in response to the request for a password, type in the password and then press a carriage return;

(6) in response to project ID, type in some acronym to identify the project (e.g., EPA or RYN) and then press a carriage return;

(7) in response to the client lock, enter the appropriate code, then press a carriage return;

(8) in response to the request for a new client lock, enter a new code if desired and then press a carriage return; if no new client lock is desired simply press a carriage return;

(9) if a new client lock was entered in step 8, the user will be asked to repeat it, after which a carriage return should be pressed;

(10) the user will now be prompted with a period and the user enters MLAB as follows:

```
.RUN PRO:MLAB
```

and then a carriage return;

(11) the user will now be asked if he wishes to keep a file of the MLAB session. If a file is desired type Y and a carriage return. If a file is not desired simply press a carriage return;

(12) the user is now in MLAB and will receive an * as a prompt for input. After each line of input the user must press a carriage return. The user will then receive another input prompt or will receive output followed by an input prompt;

(13) to leave MLAB the user types EXIT after an input prompt:

```
*EXIT
```

(14) after exiting from MLAB, the user will receive a period for a prompt and may then log off by typing either of the following:

```
.KJ or
.BYE
```

and then pressing a carriage return. With the period prompt the user may again enter MLAB by typing the RUN PRO:MLAB command or may go to the text editor as indicated in Chapter 5;

(15) to view the log-file of the MLAB session the user types:

```
.TYPE MLAB.LOG
```

To save this file so that it will not be overwritten at a later MLAB session the user must rename it by typing:

```
.COP filename.ext=MLAB.LOG
```

where 'filename' is a six-letter name and 'ext' is a three-letter extension.

(16) after logging off, remove the phone from the acoustic coupler.

In the text which follows we maintain the following conventions: if an example is labeled "EXAMPLE MLAB J" then the example will contain some statements which begin with *; these statements are actual MLAB input statements. Other statements will be MLAB output or explanatory comments. Except for the explanatory comments, these examples are exactly as they would appear in MLAB. The symbol * is an MLAB prompt for input and is not supplied by the user.

If an example is labeled "EXAMPLE K " then the example only contains input statements as they would appear in MLAB and perhaps some explanatory comments. No MLAB output is given.

I.2 ARITHMETIC IN MLAB

In its simplest form, MLAB can be use to perform elementary algebraic operations on scalars and functions. This section treats these basic operations. The symbols +,-,* and / are used by MLAB as the addition, subtraction, multiplication and division operators.

An equals sign or a back arrow is used to assign a numerical value to a variable. On some keyboards, a horizontal bar replaces the back arrow; on those keyboards the horizontal bar may be used in assignment statements. The MLAB command TYPE is used to cause a quantity to be printed out. In the first 6 examples below the desired scalar value will printed out by MLAB in response to the TYPE command. The TYPE W command in example 7 will simply cause the definition of W to be retyped.

EXAMPLE 1.

```
TYPE 24*SQRT(17)
```

EXAMPLE 2.

```
A=3;B=4
TYPE A+3*B
```

EXAMPLE 3.

A vertical arrow (a circumflex on some keyboards) or a double star is used to indicate exponentiation.

```
PI=3.14159
TYPE SIN(3*PI)+PI^2
X=3;Y=1
TYPE X**2+Y**3
```

EXAMPLE 4.

Function assignments must include the word FUNCTION (or FCT) whereas scalar assignments need only the equals or the back arrow.

```
FUNCTION F(X)=X*SIN(2*X)
TYPE F(3)
```

EXAMPLE 5.

```
FUNCTION F(X,Y)=X*EXP(-X*Y)
TYPE F(1,1)
```

EXAMPLE 6.

Function definitions may contain parameters, but these parameters must be given specified values before the function is evaluated.

```
FUNCTION F(X)=A+B*X
A=2;B=3
TYPE F(.2)
```

EXAMPLE 7.

```
FCT W=X^2*EXP(-X)+3/SQRT(1+X^2)
TYPE W
```

MLAB will prompt the user for input with an * and at the end of each line of input the user must press a carriage return.

EXAMPLE MLAB 1

```
*TYPE 14+16.2/SQRT(3)

SCALAR = 23.35308
```

EXAMPLE MLAB 2

```
*A=12;B=3
*TYPE (A/B)^2

SCALAR = 16.00000
```

EXAMPLE MLAB 3

```
*FUNCTION H(X)=(X^2)*EXP(-X)
*TYPE H(2)

SCALAR = .5413411
```

EXAMPLE MLAB 4

```
*FUNCTION F(X,Y)=A+B*X+C*Y +D*X*Y
*A=1;B=0;C=2;D=-1
*TYPE F(1,1)
```

```
SCALAR = 2.000000

-------------------------------------------------------------------------
-------------------------------------------------------------------------
```

I.3 ELEMENTARY MATRIX OPERATIONS

There are several different methods for entering data into a matrix; most of these methods involve entering the data column by column. We now give some typical examples.

EXAMPLE 1.

```
M COL 1=LIST(1,2,3,4,5,6)
M COL 2=1:6
M COL 3=2:12:2
M COL 4=READ("TTY:")
```

In the above example M COL J specifies the column of M into which we wish to place data. The first command above uses the MLAB LIST operator which is followed by the data separated by commas and enclosed in parentheses. The LIST command, in this case, simply tells MLAB to place in Column 1 of the matrix M the values 1,2,3,4,5, and 6. Data is entered in the second and third column via the MLAB convention that uses a:b:c to represent a column vector starting with a and incrementing by c until b is reached; if c is omitted then an increment of 1 is understood.

The READ("TTY:") command above returns control to the keyboard and the data is entered on the keyboard separated by commas or blanks and after the last value is typed the MLAB user types a control Z (that is, depress the CTRL and Z keys at the same time).

Assuming that 6 values are entered in M COL 4, the statements in EXAMPLE 1 create a 6 by 4 matrix named M. the command TYPE M would cause the matrix M to be typed out. By using the ROW and COL MLAB operators, the user can type out the rows and columns of M.

EXAMPLE 2.

```
TYPE M
TYPE M COL 2
TYPE M COL 1
TYPE M ROW(2,3)
```

EXAMPLE 3.

```
TYPE M,M COL 2,M COL 1,M ROW(2,3)
```

EXAMPLE 4.

```
TYPE M; TYPE M COL 2;TYPE M COL 1;TYPE M ROW(2,3)
```

EXAMPLES 2,3, and 4 yield the same output.

If we wish to change the second element in the first row of the matrix M from 2 to .5 it suffices to write

EXAMPLE 5.

```
M(1,2)=.5
```

In general, N(I,J) represents the Jth element in the Ith row of the matrix N.

EXAMPLE 6.

```
K COL 1=LIST(-1,2,3)
K COL 2=LIST(4,1)
```

The matrix K defined in EXAMPLE 6 has columns of unequal length; MLAB will automatically add a zero as the third entry in column 2 so that the columns will be compatible in length.

If M and N are matrices and c is a scalar, then we can perform the following operations in MLAB:

```
M+N, M*N, c*M, M+c
```

and if c is an integer we can also compute M^c.

If two matrices are not of the appropriate size for a given algebraic operation, MLAB will add zero entries until the matrices are compatible in size and then will perform the indicated algebraic operation.

The transpose of a matrix N is obtained by interchanging the rows and the columns of the matrix N. An apostrophe is used in MLAB to obtain the transpose of a matrix.

EXAMPLE 7.

```
TYPE M'
```

We use the letter I to denote the identity matrix, that is the matrix with ones down the diagonal and zeros elsewhere. Given a matrix A, the matrix B with the property that

```
A*B = B*A = I
```

is called the inverse of A. The inverse is computed by raising A to the power -1; i.e., B=A^-1.

EXAMPLE 8.

```
A COL 1=LIST(1,2)
A COL 2=LIST((-1,4)
B=A^-1
TYPE B, B*A, A*B
```

The last two matrices typed out in EXAMPLE 8 are equal to the identity matrix.

The matrix M will be the same in the following 6 examples.

EXAMPLE MLAB 1

The following MLAB dialog illustrates the use of some of the above-mentioned matrix instructions of MLAB. We start by building a matrix M column by column. M COL 1 refers to the first column of M, M COL 2 the second, etc. The expression a:b is a column matrix whose first element is a, second is a+1 and so on. The last element is less than or equal to b.

The expression a:b:c is a column matrix whose first element is a, second is a+c etc., and whose last element is less than or equal to b, if c is positive, and greater than or equal to b, if c is negative.

The LIST operator returns a column matrix whose elements are the arguments of the LIST. The : operator may be used within a LIST expression. The matrix M, which we type out, is obtained using the four commands shown.

```
*M COL 1=1:4
*M COL 2=2:3.5:.5
*M COL 3=LIST(7,2,4,1)
*M COL 4=LIST(1,6:8)
*TYPE M

MATRIX M:

1.000000 2.000000 7.000000 1.000000 *
2.000000 2.500000 2.000000 6.000000 *
3.000000 3.000000 4.000000 7.000000 *
4.000000 3.500000 1.000000 8.000000 *
```

EXAMPLE MLAB 2

The ´ operator returns the transpose of a matrix.

```
*TYPE M´

MATRIX :

1.000000 2.000000 3.000000 4.000000 *
2.000000 2.500000 3.000000 3.500000 *
7.000000 2.000000 4.000000 1.000000 *
1.000000 6.000000 7.000000 8.000000 *
```

EXAMPLE MLAB 3

We may use the list and : operators along with the transpose operator ', to build a matrix row by row, as the following example illustrates.

```
*O ROW 1=(1:4)'
*O ROW 2=LIST(1,2,5,3)'
*TYPE O

MATRIX O:

1.000000 2.000000 3.000000 4.000000 *
1.000000 2.000000 5.000000 3.000000 *
```

EXAMPLE MLAB 4

The COL or ROW operators may be used to reference specified column(s) or row(s) of a matrix. The : operator may be used within a COL or ROW expression.

```
*TYPE M COL(2,4)

MATRIX :

2.000000 1.000000 *
2.500000 6.000000 *
3.000000 7.000000 *
3.500000 8.000000 *

*TYPE M ROW(2:4)

MATRIX :

2.000000 2.500000 2.000000 6.000000 *
3.000000 3.000000 4.000000 7.000000 *
4.000000 3.500000 1.000000 8.000000 *
```

EXAMPLE MLAB 5

Most of the same arithmetic operators which apply to scalars may also be applied to matrices. The sum of two matrices of equal size is the matrix obtained by adding the corresponding entries in the two matrices. The product of two matrices is the standard matrix product.

Adding a scalar to a matrix simply adds that scalar to each entry in the matrix; multiplying a matrix by a scalar simply multiplies each entry in the matrix by that scalar.

```
*TYPE M*M

MATRIX :

30.00000 31.50000 40.00000 70.00000 *
37.00000 37.25000 33.00000 79.00000 *
49.00000 50.00000 50.00000 105.0000 *
46.00000 47.75000 47.00000 96.00000 *

*TYPE M*.3

MATRIX :

.3000000 .6000000 2.100000 .3000000 *
.6000000 .7500000 .6000000 1.800000 *
.9000000 .9000000 1.200000 2.100000 *
1.200000 1.050000 .3000000 2.400000 *

*TYPE M+100

MATRIX :

101.0000 102.0000 107.0000 101.0000 *
102.0000 102.5000 102.0000 106.0000 *
103.0000 103.0000 104.0000 107.0000 *
104.0000 103.5000 101.0000 108.0000 *

*TYPE M+M*M

MATRIX :

31.00000 33.50000 47.00000 71.00000 *
39.00000 39.75000 35.00000 85.00000 *
52.00000 53.00000 54.00000 112.0000 *
50.00000 51.25000 48.00000 104.0000 *
```

EXAMPLE MLAB 6

Matrices may also be raised to powers, provided that the exponent is a scalar and an integer. The matrix M^2 is equal to M*M and M^3 equals M*M*M. We see from the following dialog that the inverse of M multiplied by M yields the identity matrix of the same size as M and that raising the inverse to a positive power yields the same result as raising the matrix to the same negative power.

```
*TYPE M^2

MATRIX :

30.00000 31.50000 40.00000 70.00000 *
37.00000 37.25000 33.00000 79.00000 *
49.00000 50.00000 50.00000 105.0000 *
46.00000 47.75000 47.00000 96.00000 *
```

```
*TYPE M^3

MATRIX :

493.0000 503.7500 503.0000 1059.000 *
526.5000 542.6250 544.5000 1123.500 *
719.0000 740.5000 748.0000 1539.000 *
666.5000 688.3750 701.5000 1429.500 *

*N=M^-1
*TYPE N

MATRIX N:

-.8333336@-1 -1.416667     .7500000     .4166667     *
 .6666667     1.333333    -2.000000     .6666667     *
 .1862645@-8 -.2000000     .4000000    -.2000000     *
-.2500000     .1500000     .4500000    -.3500000     *

[The @ notation above represents powers of ten.]

*TYPE N*M

MATRIX :

 1.000000    -.298ø232@-7 -.1378357@-6 -.8940697@-7  *
 .1192093@-6  1.000000     .1043081@-6  .3576279@-6  *
-.2980232@-7 -.3725290@-7  1.000000    -.8940697@-7  *
-.4470348@-7 -.2980232@-7  .2235174@-7  .9999999     *

*TYPE M*N

MATRIX :

 1.000000    -.1359731@-6  .1676381@-6 -.6705523@-7  *
-.1490116@-7  .9999999     .8940697@-7  .0000000     *
-.1490116@-7 -.2086163@-6  1.000000    -.2980232@-7  *
-.4470348@-7 -.2235174@-6  .1788139@-6  1.000000     *

*TYPE N^3

MATRIX :

-.8333333    -1.845000     3.665000    -1.878333     *
 .5000000     1.566667    -2.633333     1.233333     *
-.2083333    -.4996667     .8143334    -.3296667     *
 .2500000     .3510000    -.8403333     .4443333     *

*TYPE M^-3

MATRIX :

-.8333333    -1.845000     3.665000    -1.878333     *
 .5000000     1.566667    -2.633333     1.233333     *
-.2083333    -.4996667     .8143334    -.3296667     *
 .2500000     .3510000    -.8403333     .4443333     *
```

EXAMPLE MLAB 7

This example illustrates the use of MLAB for the solution of a system of simutaneous linear equations using the MLAB matrix-inversion operator. Consider the following system of simultaneous equations

```
X+Y+Z=1
2X+3Y+3Z=4
3X+7Y-3Z=6
```

From linear algebra we know that if we represent a system of simultaneous linear equations in the form MA=B, where M is the matrix of the coefficients of the unknowns, A=(X,Y,Z) is the vector of unknowns and B is the vector of values on the right-hand side of the equations, then the solution of the system is given by A=M^-1*B, where the matrix M^-1 is the inverse of the matrix M.

Our first step is to enter the matrix of coefficents and the vector of the right-hand side values into MLAB.

```
*M COL 1=LIST(1,2,3)
*M COL 2=LIST(1,3,7)
*M COL 3=LIST(1,3,-3)
*B=LIST(1,4,6)
```

We use the MLAB matrix-inversion operator to determine the inverse of M, and then multiply this by the vector of the right-hand side, B, to obtain the solution of the system.

```
*O=M^-1
*TYPE M

MATRIX M:

 1.000000      1.000000      1.000000      *
 2.000000      3.000000      3.000000      *
 3.000000      7.000000     -3.000000      *

*TYPE O

MATRIX O:

 3.000000     -.9999999      .1885928@-7   *
-1.500000      .6000000      .9999999@-1   *
-.5000000      .4000000     -.1000000      *

*A=O*B
*TYPE A
```

```
MATRIX A:

-.9999998 *
 1.500000 *
 .5000000 *
```

We see that the solution of the system of equations is X=-1, Y=1.5, and Z =.5. To check the solution we define the functions F,G and H as follows:

```
*FCT F(X,Y,Z)=X+Y+Z
*FCT G(X,Y,Z)=2*X+3*Y+3*Z
*FCT H(X,Y,Z)=3*X+7*Y-3*Z
```

We now evaluate F,G and H at (-1,1.5,.5) and compare with the original values 1,4, and 6 which formed the right hand side of our original system of equations.

```
*TYPE F(-1,1.5,.5),G(-1,1.5,.5),H(-1,1.5,.5)

SCALAR = 1.000000

SCALAR = 4.000000

SCALAR = 6.000000
```

We see that MLAB did, in fact, produce the solution to the given system of equations.

I.4 THE POINTS OPERATOR AND THE DRAW COMMAND

If M is a two column matrix, then the command DRAW M followed by DISPLAY=T4010 (if the user is on a TEKTRONIX 4010) will cause a solid curve to be drawn through the points in M and to be displayed on the screen. If the user is on a TEKTRONIX 4012, 4014, or 4051, then the appropriate commands would be DISPLAY=T4012, etc.

EXAMPLE 1.

```
M COL 1=LIST(1,2,3,4,5)
M COL 2=LIST(1.1,2.4,3.5,4.6,5.9)
DRAW M
DISPLAY=T4010
```

The result of the commands in EXAMPLE 1 will be a solid curve through the points (1,1.1), (2,2.4),...,(5,5.9).

The points (1.1.1), (2, 2.4), ..., (5, 5.9) will not be marked in any way on the graph if we use only the DRAW command. By using the POINTTYPE command we can highlight the data points in M.

EXAMPLE 2.

```
M COL 1=LIST(1,2,3,4,5)
M COL 2=LIST(1.1,2.4,3.5,4.6,5.9)
DRAW M,POINTTYPE 3
DISPLAY=T4010
```

The result of EXAMPLE 2 will be the same as EXAMPLE 1 with the exception that the data points will be marked with small triangles. The POINTTYPE command can be followed by any of the numbers 1 through 6 or by a single character enclosed in quotes such as "X". The codes for the various pointtypes are listed in the appendix.

EXAMPLE 3.

```
M COL 1=LIST(1,2,3,4,5)
M COL 2=LIST(1.1,2.4,3.5,4.6,5.9)
DRAW M, POINTTYPE "A"
DISPLAY=T4010
```

The data points in EXAMPLE 3 will be labeled with an A and will be connected by a solid line. MLAB can connect data points by several different types of curves; the LINETYPE command is used to specify the type of line (curve) that is to be used in joining data points.

EXAMPLE 4.

```
M COL 1=1:5
M COL 2=LIST(1.1,2.4,3.5,4.6,5.9)
DRAW M,POINTTYPE 3,LINETYPE 2
```

```
DISPLAY=T4010
```

The data points in EXAMPLE 4 will be marked with small triangles and will be connected with a dashed curve rather than by a solid curve. The command LINETYPE can be followed by any of the numbers 1 through 6. The codes for the various linetypes are listed in the appendix.

To obtain the graph of a function F(X), we first specify a matrix of points which lie on the graph of F

```
X1 F(X1)
X2 F(X2)
. .
. .
. .
XN F(XN)
```

and then we draw a smooth curve through these points. The MLAB operator POINTS forms a matrix of the above type for a given function F(X).

EXAMPLE 5.

```
FUNCTION F(X)=X*SIN(X)
L=POINTS(F,0:1:.1)
DRAW L
DISPLAY=T4010
```

The second argument of the POINTS operator is a vector of values at which F is to be evaluated. The rows of the matrix L in EXAMPLE 5 are

```
0 F(0)
.1 F(.1)
.2 F(.2)
. .
. .
. .
1 F(1)
```

The result of EXAMPLE 5 will be a smooth curve through the points (0, F(0)), (.1, F(.1)), (.2, F(.2)), ..., (1, F(1)).

EXAMPLE 6.

```
FUNCTION G(X)=X*SIN(X)
V=0:1:.1
K=POINTS(F,V)
```

The matrix K in EXAMPLE 6 is the same as the matrix L in EXAMPLE 5.

The POINTS operator is not restricted to functions of one variable, but can be used for functions defined on any number of variables.

EXAMPLE 7.

```
FUNCTION H(X,Y)=X+EXP(X-Y)
W COL 1=LIST(1,2,3,4,5,6)
W COL 2=LIST(1,2,3,4,5,6)
Z=POINTS(H,W)
TYPE Z
```

The matrix Z in EXAMPLE 7 will have three columns, the first two columns will contain X,Y values and the third column will contain the corresponding H(X,Y) values.

EXAMPLE 8.

```
FUNCTION A(X,Y)=X^2-Y^2
FUNCTION B(X)=3*X^2+2
C=POINTS(B,1:2:.1)
D=POINTS(A,C)
TYPE D
```

The matrix D in EXAMPLE 8 has three columns corresponding to the portion of A above the parabola defined by B.

EXAMPLE MLAB 1

This example exhibits the use of the MLAB DRAW command in its simplest form. If M is a two column matrix then the commands

```
DRAW M
DISPLAY=T4012
```

will cause a solid curve to be drawn through the points (X1,Y1),...,(XN,YN) which form the rows of M. There is an automatic frame of reference, called the DEFAULTWINDOW, which constructs the axes and labels them for the curves which are being drawn. The user has the option of building his own window and labelling the axes as he wishes.

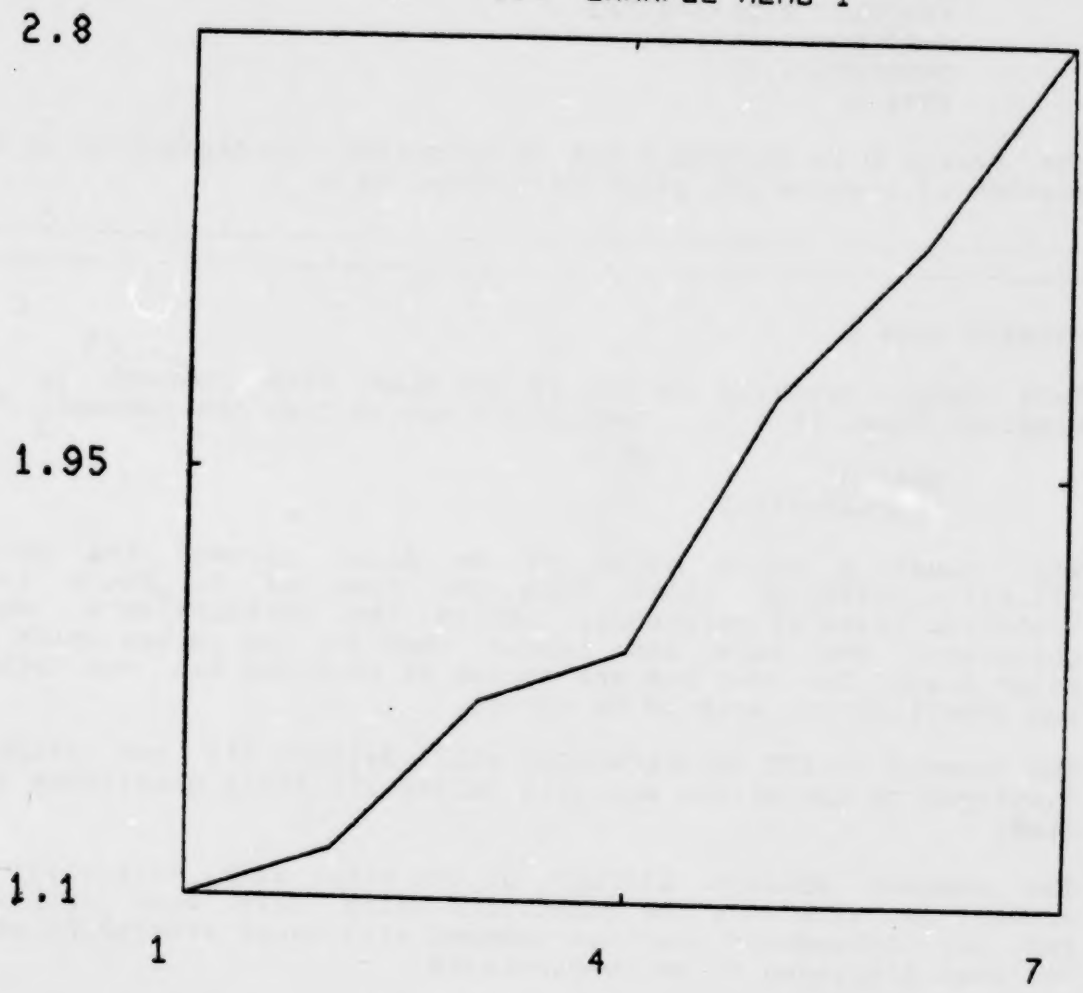
The command DELETE DEFAULTWINDOW will delete all the graphics displayed on the screen and will delete all these quantities from MLAB.

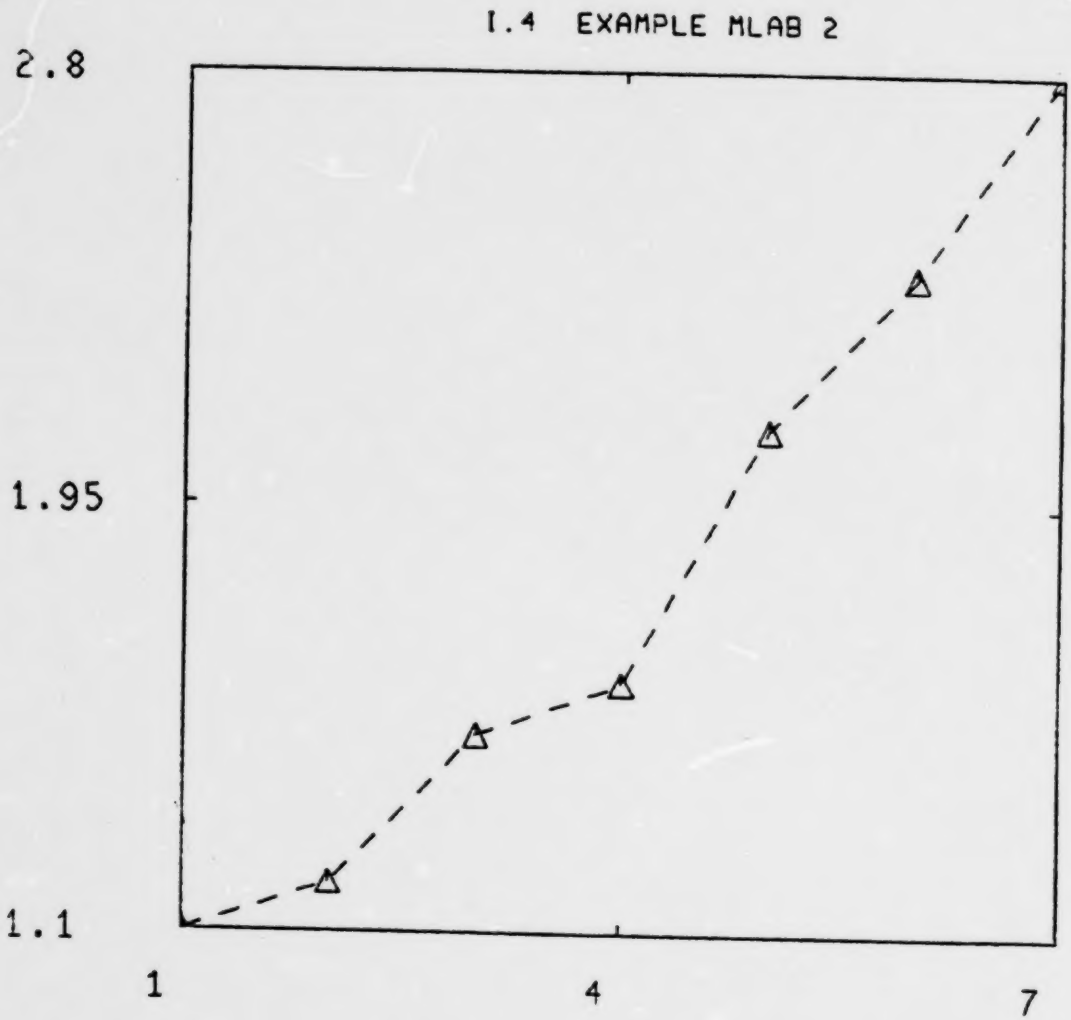
The command RELEASE DISPLAY, on the other hand, will clear the screen, but will save the quantities which have been displayed and any subsequent graphics command will cause everything which had been displayed to be re-displayed.

The command GET DISPLAY after a RELEASE DISPLAY will return the display to the screen.

```
*M COL 1=1:7
*M COL 2=LIST(1.1,1.2,1.5,1.6,2.1,2.4,2.8)
*DRAW M

*DISPLAY=T4012

*DELETE DEFAULTWINDOW
*EXIT
```

I.4 EXAMPLE MLAB 1




1.4 EXAMPLE MLAB 2
2.8
1.95
1.1
1
4
7

EXAMPLE MLAB 2

In this example we use the POINTTYPE and LINETYPE commands; the codes for these commands are at the end of the APPENDIX. If no LINETYPE is specified in the DRAW command, a solid curve will be drawn through the data points as was the case in the preceding example. If no POINTTYPE is specified then the data points on the curve will not be highlighted in any way.

```
*M COL 1=1:7
*M COL 2=LIST(1.1,1.2,1.5,1.6,2.1,2.4,2.8)
*DRAW M,POINTTYPE 3,LINETYPE 2
*DISPLAY=T4012
*RELEASE DISPLAY
*EXIT
```

EXAMPLE MLAB 3

In this example we use the MLAB STRING command to place a verbal string in the graph. Every quantity which is displayed graphically has a name. In this example, we give the command DRAW M without assigning any name to the curve which will be drawn through the points in M. MLAB will assign the default name TEMPC1 to that curve. If we used the command

```
DRAW AP=M
```

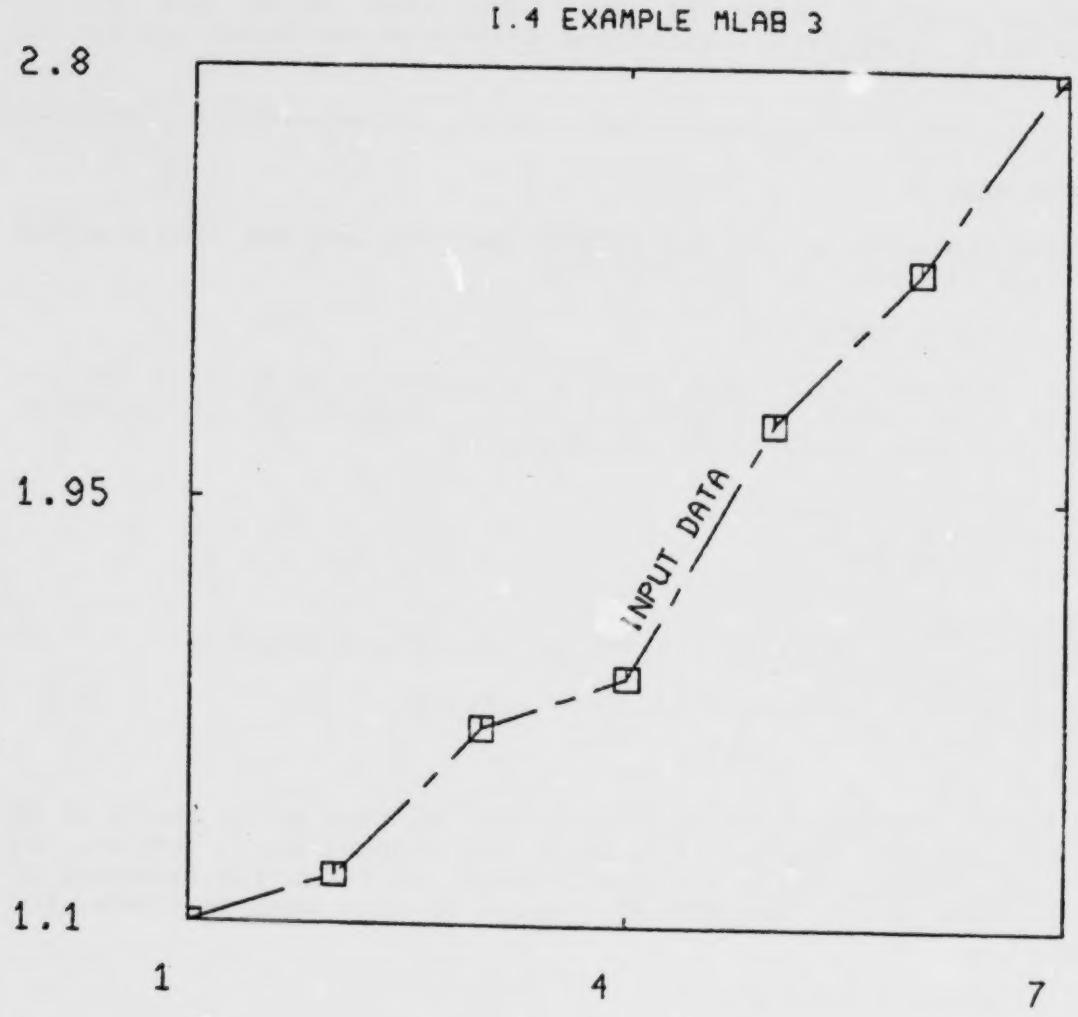
then the curve through M would have the name AP.

In this example the name C1 is explicitly given to the text to be incorporated in the graph via the STRING command. The text is placed inside quotation marks. The starting point for the text is specified by the AT clause; the angle from the horizontal at which the text is to be displayed is included in the ANGLE clause.

The AT, SIZE, and ANGLE clauses may be omitted in the STRING command, in which case they will receive the default values (0,0), 5 and 0.

```
*M COL 1=1:7
*M COL 2=LIST(1.1,1.2,1.5,1.6,2.1,2.4,2.8)
*DRAW M,POINTTYPE 4,LINETYPE 4

*STRING C1="INPUT DATA",AT 4,1.7,SIZE 3,ANGLE 60
*DISPLAY=T4012
*DELETE DEFAULTWINDOW

*EXIT
```

If the text included in the graph does not appear as the MLAB user wishes it to appear then the command


I.4 EXAMPLE MLAB 3
2.8
1.95
1.1
1
4
7
INPUT DATA

```
DELETE C1
```

will cause the verbal text in C1 to be deleted from the graphic display and nothing else will be deleted. The user may then repeat the STRING command with the appropriate modifications.

On the other hand, the command

```
STRING C1,SIZE 5
```

will retain all the information contained in C1 except the size which will be changed to 5. In this way, any of the attributes of the given string C1 can be changed.

Giving an explicit name to the quantities which are to be graphically displayed facilitates deletions and modifications of these quantities.

EXAMPLE MLAB 4

In this example we use the POINTS operator and the DRAW command to obtain the graph of the function

```
F(X)=X*EXP(-X^2)
```

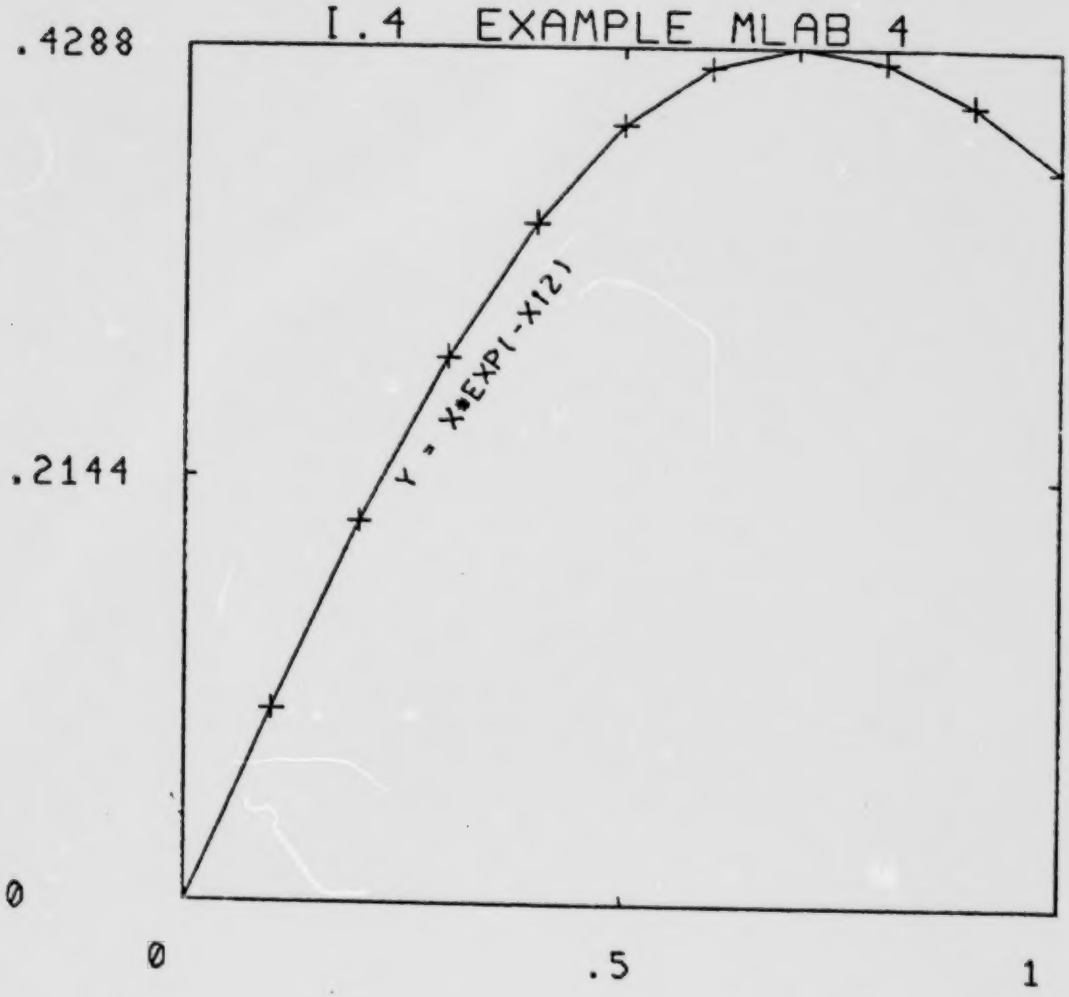
on the interval [0,1]. Note that a circumflex can be used for exponentiation rather than an upper arrow. Usually one or the other of these symbols will be on the keyboard.

```
*FUNCTION F(X)=X*EXP(-X^2)
*N=POINTS(F,0:1:.1)
*DRAW N,POINTTYPE 2

*DISPLAY=T4012
*STRING A1="Y = X*EXP(-X^2)",AT .25,.21,SIZE 3,ANGLE 50
*STRING A1,ANGLE 55
*STRING B1="I.4 EXAMPLE MLAB 4",AT .15,.44

*EXIT
```

The first STRING command positioned the text at an angle of 50 degrees with the X-axis; this angle was changed to 55 degrees by the next command. Only the final result of those two commands is shown on the graph. The STRING command is also used to place the title

```
EXAMPLE MLAB 4
```

on the top of the graph. The default size (5) and angle (0) were used in this STRING command.

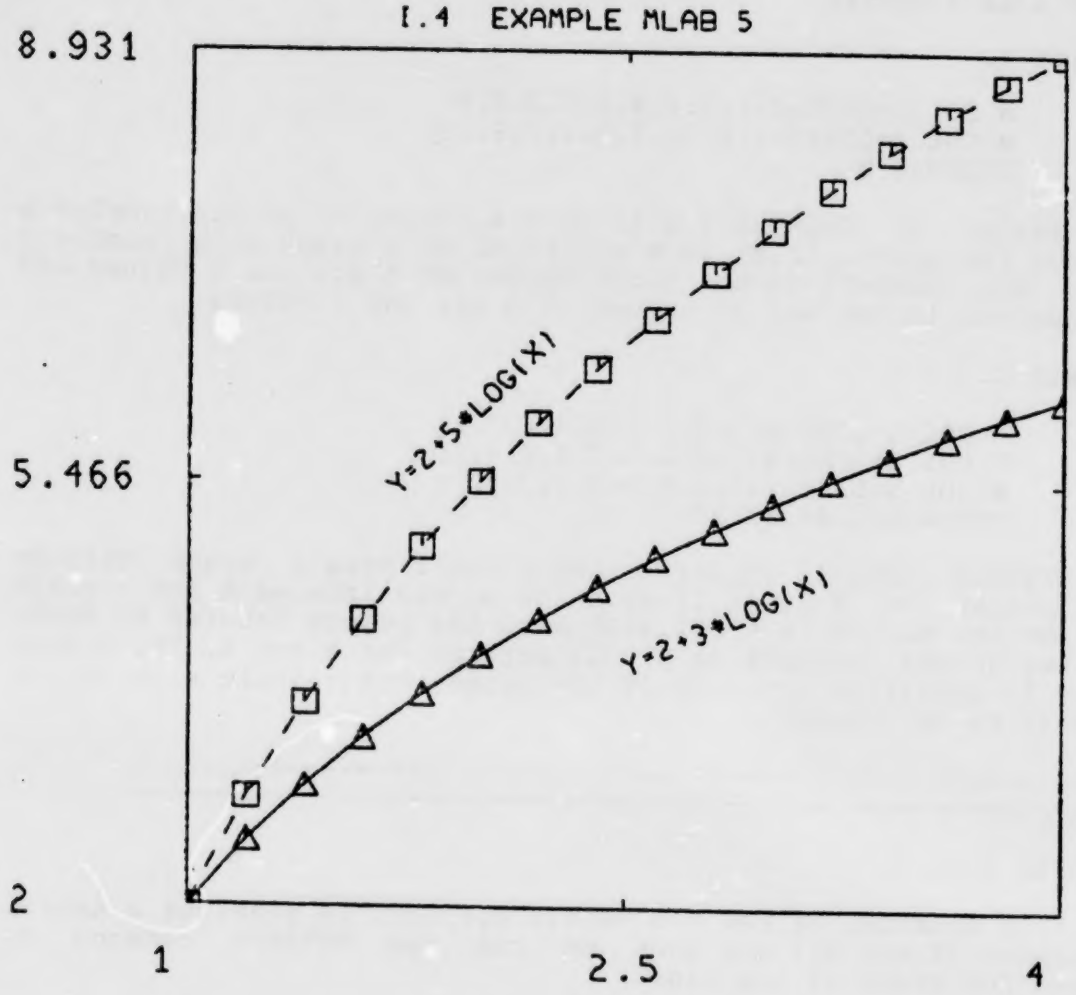

I.4 EXAMPLE MLAB 4
.4288
.2144
0
0
.5
1
Y = X*EXP(-X12)

EXAMPLE MLAB 5

In this example we use the POINTS operator to generate the graphs of functions and we use the POINTTYPE and LINETYPE operators to simplify the comparison of these two curves.

The STRING operator is used to label the curves and to give a title to the graph.

```
*FUNCTION K(X)=2+3*LOG(X)
*FUNCTION L(X)=2+5*LOG(X)
*V=1:4:.2

*N1=POINTS(K,V)
*N2=POINTS(L,V)

*DRAW N1,POINTTYPE 3

*DRAW N2,POINTTYPE 4,LINETYPE 2

*DISPLAY=T4012

*STRING A1="Y=2+5*LOG(X)",AT 1.7,5.4,SIZE 3,ANGLE 45

*STRING B1="Y=2+3*LOG(X)",AT 2.5,4,SIZE 3,ANGLE 25

*STRING C1="I.4 EXAMPLE MLAB 5",AT 1.7,9.2,SIZE 3

*EXIT
```



I.4 EXAMPLE MLAB 5
8.931
5.466
2
1
2.5
4
Y=2.5*LOG(X)
Y=2.3*LOG(X)

I.5 THE TTYDRAW COMMAND

In addition to the DRAW command, MLAB possesses another command with a graphics capability - the TTYDRAW command. One advantage of the TTYDRAW command is that it may be used on a teletype terminal as well as on a CRT terminal.

The basic form of the command is

```
TTYDRAW M
```

where M is a matrix.

EXAMPLE 1.

```
M COL 1=LIST(1.1,1.4,2.3,2.7,2.8)
M COL 2=LIST(-3.1,-2.7,3.1,1.8,.2)
TTYDRAW M
```

The result of EXAMPLE 1 will be a sequence of points labeled A through the points (X,Y) in M displayed on a graph with numbered axes. The numbers in the first column of M are the X values and the numbers in the second column of M are the Y values.

EXAMPLE 2.

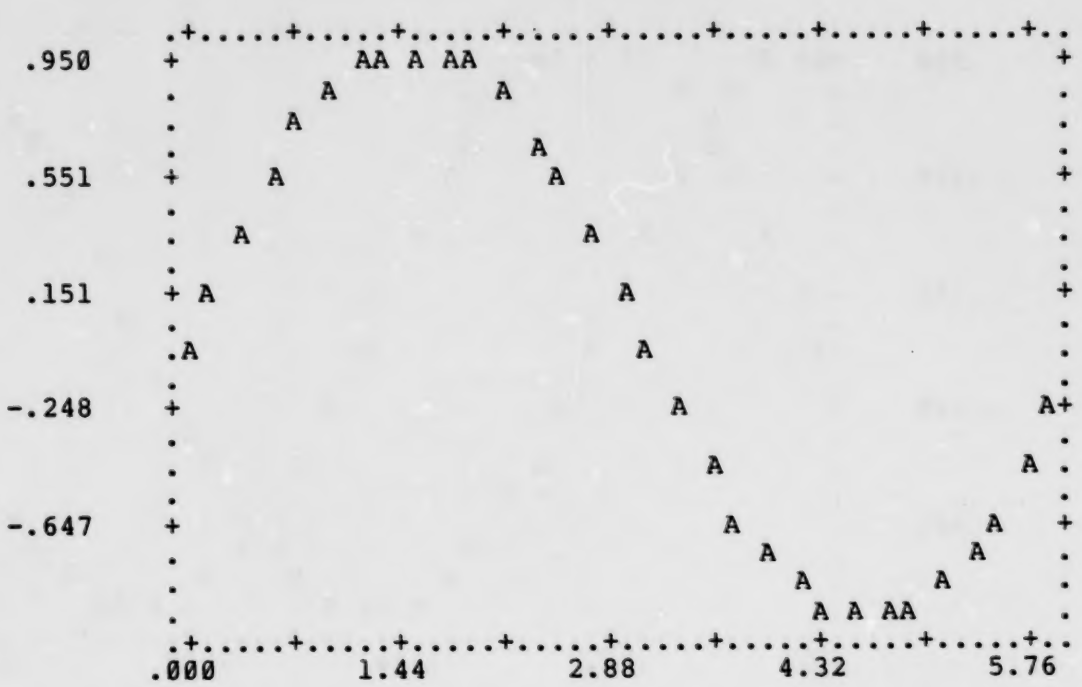
```
K COL 1=LIST(4.3,4.5,3.9,5.6,7.1)
K COL 2=LIST(4.1,4.2,3.7,3.8,4.1)
K COL 3=LIST(2.7,2.9,2.3,3.7,3.5)
TTYDRAW K,40 BY 15
```

The TTYDRAW command, in this example, will draw a graph through the points in K COL(1,2) with the points labeled A and a graph through the points in K COL(1,3) with the points labeled B. Where the two graphs coincide an * will replace the A and B. The clause 40 by 15 specifies the size of the graph; the default size is 60 (lines) by 20 (lines).

EXAMPLE MLAB 1

In this example, we use the POINTS operator to generate a matrix of values (X,sin(X)) and then we use the TTYDRAW command to obtain the graph of the sine.

```
*M=POINTS(SIN,0:6:.2)
*TTYDRAW M,50 BY 20
```

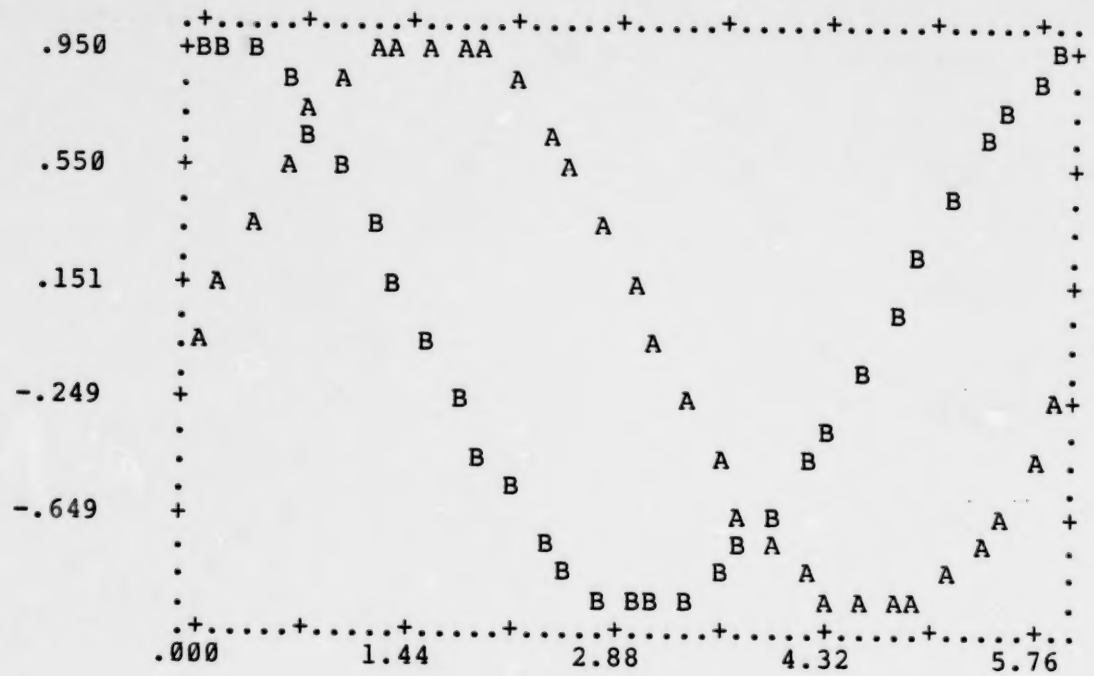


EXAMPLE MLAB 2

In this example, we use the matrix M from the preceding example and we use the POINTS operator to obtain a matrix of values for the cosine. We join the matrices M and N to form a matrix K whose first two columns give (X,sin(X)) values and whose first and third columns contain the (X,cos(X)) values.

Note that K=M&'(N COL 2), where &' is the operator which concatenates columns of matrices.

```
*N=POINTS(COS,0:6:.2)
*K COL 1=M COL 1
*K COL 2=M COL 2
*K COL 3=N COL 2
*TTYDRAW K,50 BY 20
```



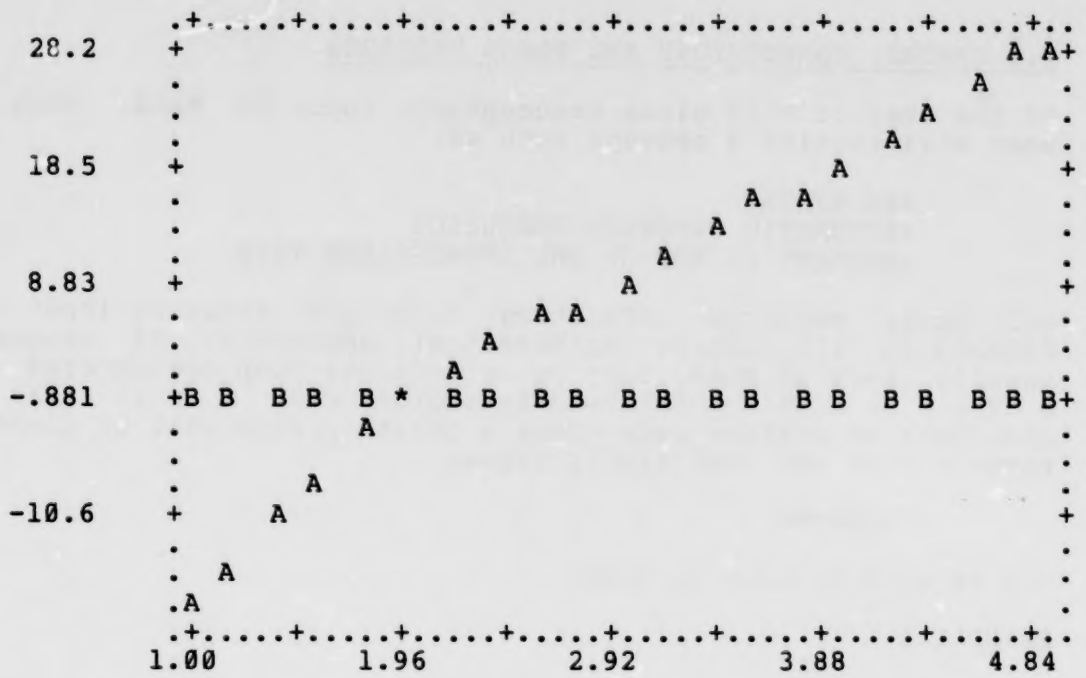
EXAMPLE MLAB 3

In this example we will use the TTYDRAW command to give us a rough approximation to a root of the equation

$$X^2 + 3*LOG(X) = EXP(4-X)$$

We will graph the function X^2+3*log(X)-exp(4-X) and the function Y=0. The roots of the equation occur at the points where the two graphs cross. If the TTYDRAW command is being used to graph several curves then an * will mark any points where the graphs cross.

```
*FUNCTION Z(X)=X^2+3*LOG(X)-EXP(4-X)
*W=POINTS(Z,1:5:.2)
*W COL 3=0
*TTYDRAW W,50 BY 20
```



We can see graphically that there is a root at approximately X=2. To find the root exactly we can use the MLAB ROOT operator (see Chapter IV).

I.6 ERRORS, CORRECTIONS AND ERROR MESSAGES

If the user of MLAB gives unacceptable input to MLAB, then the user will receive a message such as:

```
BAD SYNTAX
ARITHMETIC OVERFLOW/UNDERFLOW
ARGUMENT IS NOT OF THE APPROPRIATE TYPE
```

and other messages indicating incorrect input or input which demands an ill-defined mathematical operation. If things go totally awry in MLAB, such as an infinite loop encountered, then a CTRL C or a CTRL C followed by another CTRL C will take the user back to monitor mode where a period prompt will be given. To reenter MLAB the user simply types:

```
.REENTER
```

and he will be back in MLAB.

EXAMPLE 1.

```
A=3;B=4
TYPPE A^2+4*B
```

In this example a simple typographical error was made. If the error is detected before the carriage return is hit, the user may simultaneously backspace and erase by using either a CTRL H or a RUBOUT and then may enter the appropriate text.

EXAMPLE 2.

```
A=3;B=4
TYPPE A^2+4*B/B*4+2^A EP/E A^2+4*B
```

In this example the RUBOUT key was pressed for each character from B back to the second P in TYPPE. Then the correct text was entered. The slashes are automatically produced by the RUBOUT; the user does not supply them.

Pressing a control H causes the pointer to physically move backwards as the deletions are being made, whereas pressing the RUBOUT causes the pointer to move forward.

EXAMPLE 3.

```
A=3;B=4
TYPPE A^2+4*B DELETED
TYPE A^2+4*B
```

In this example instead of deleting all the characters after the error and then retyping, a CTRL U was pressed after the B. The current line of input is completely deleted by the CTRL U and a feed is made to the next line and input is awaited; an input prompt is not given in this situation.

EXAMPLE MLAB 1

```
*TYPE17+LOG(4)
BAD SYNTAX! (ERROR#7)
*TYPE 17+LOG(4)

SCALAR = 18.38629
```

In this example we neglected to leave a space between TYPE and 17. If we had detected the error before hitting the carriage return we could have made the correction using the CTRL H or the RUBOUT. The error went undetected, however, and the carriage return was pressed. The result is an error message from MLAB followed by an input prompt.

After an error message or a sequence of error messages MLAB will give the user an input prompt. The one exception is the situation in which an infinite loop is generated; in this case the user must use the CTRL C command twice to exit from the loop.

EXAMPLE MLAB 2

```
*FUNCTION F(X)=X*SIN(X)
*TYPE F(PI/2)
ARGUMENT IS NOT OF THE APPROPRIATE TYPE OR FORM!
*PI=3.14159
*TYPE F(PI/2)

SCALAR = 1.570795
```

In this example we attempted to evaluate the function F at a quantity which was not a number and received an error message.

EXAMPLE MLAB 3

```
*FUNCTION F(X)=X^2-X^3
F WAS IN USE. IT HAS BEEN REDEFINED.
*TYPE F

FUNCTION F(X) = X^2-X^3
```

In this example we defined a function F(X), but F had been used in EXAMPLE MLAB 2. MLAB informed us that F had been in use and has been redefined.

EXAMPLE MLAB 4

```
*M COL1=LIST(1,2,3)
BAD SYNTAX! (ERROR#11)

*M COL 1=LIST(1,2,3)
*M COL(2)=LIST(4,5,6)
*M COL 3=LIST(7,8,9)
*TYPE M

MATRIX M:

1.000000 4.000000 7.000000 *
2.000000 5.000000 8.000000 *
3.000000 6.000000 9.000000 *
```

In this example a space was not provided between COL and 1. MLAB informed us that the syntax of the input was incorrect. Note that either of the notations

COL 2 OR COL(2)

are acceptable.

EXAMPLE MLAB 5

```
*M COL 1=LIST(-1,-2)
*M COL 2=LIST(-3,-4)
*TYPE M

MATRIX M:

-1.000000 -3.000000 7.000000 *
-2.000000 -4.000000 8.000000 *
 3.000000  6.000000 9.000000 *

*DELETE M
*M COL 1=LIST(-1,-2)
*M COL 2=LIST(-3,-4)
*TYPE M

MATRIX M:

-1.000000 -3.000000 *
-2.000000 -4.000000 *
```

In this example we defined M to be a 2 by 2 matrix. But M had been defined as a 3 by 3 matrix in the preceding example. MLAB will not inform us if a matrix or scalar name had previously been in use; it will simply overwrite preexisting quantities according to current instructions. In this case an extra column and row were retained in the matrix M.

The DELETE M command deletes the contents of M and frees M for further use.

EXAMPLE MLAB 6

```
*FUNCTION G(X)=1/X
*TYPE G(0)
DIVISION BY ZERO IN G

SCALAR = .1701412@39
```

In this example we instructed MLAB to divide by zero and so received a message to that effect from MLAB and an answer of plus infinity.

EXAMPLE MLAB 7

```
*FUNCTION H(X)=X-LOG(X-4)
*TYPE H(3)
LOG: Negative argument -- real part returned

SCALAR = 3.000000
```

In this example we attempted to evaluate the LOG at a negative number and received a message to that effect. MLAB then printed out the real part of the complex-valued logarithm.

EXAMPLE MLAB 8

```
*K(X)=2*X+7
K IS NOT OF THE APPROPRIATE TYPE OR FORM!
```

In this example we neglected to include the modifier FCT for the function K(X).

EXAMPLE MLAB 9

```
*FUNCTION L(Y)=Y^2-Y
*TYPE L(M)
Exponentiation under or overflow
ARITHMETIC UNDER/OVER FLOW IN L

SCALAR = .0000000
```

In this example we attempted to evaluate a function at a matrix value (M is the matrix of EXAMPLE MLAB 5 above).

EXAMPLE MLAB 10

```
*TYPE M^2/M
ARGUMENT IS NOT OF THE APPROPRIATE TYPE OR FORM!
CLEANING UP A TEMP
```

In this example we attempted to divide by a matrix.

CHAPTER II: ELEMENTARY CURVE FITTING

MLAB uses the FIT command to fit a model to a given data matrix. The curve fitting routine is a Marquardt-Levenberg routine.

EXAMPLE 1.

```
M COL 1=LIST(1,2,3,4,5)
M COL 2=LIST(-1.2,-.3,1.5,2.4,3.1)
FUNCTION F(X)=A*X+B
A=1;B=1
FIT(A,B),F TO M
```

In the above example, the matrix M is the matrix of input data and we are trying to fit a straight line specified by F(X)=A*X+B to the data. The curve fitting routine will find the best values (in the least squares sense) for the parameters A and B. Since the curve fitting routine is an iterative process, we must supply a starting point, that is, initial values for the parameters A and B.

EXAMPLE 2.

```
N COL 1=0:3:.5
N COL 2=LIST(1,1.2,1.4,1.9,2.3,1.8,1.5)
FUNCTION G(X,Y)=A+B*X+C*X^2
A=1;B=-1;C=1
FIT(A,B,C),G TO N
```

In EXAMPLE 2, we are trying to find the best quadratic fit to the input data N. After the FIT command in both EXAMPLE 1 and EXAMPLE 2, MLAB will ask the user three questions:

(1) WHAT ARE THE CONSTRAINTS ON THE PARAMETERS?(NONE)

(2) WHAT IS THE MAXIMUM NUMBER OF ITERATIONS THAT YOU WISH PERFORMED BEFORE THE ITERATION PROCESS TERMINATES?(3)

(3) HOW SMALL DO WISH THE RATIO OF SUCCESSIVE SUMS OF SQUARES TO BE, SO THAT THE PROCESS MAY BE TERMED CONVERGENT?(.001)

The expressions in parentheses are default responses; if the user presses the carriage return in response to the question then the default value is assumed.

EXAMPLE 3.

```
L COL 1=LIST(1,1,2,2,3,3,4,4)
L COL 2=LIST(1,2,1,2,1,2,1,2)
L COL 3=LIST(1.1,2.1,2.3,2.5,2,2.6,3.4,3.8)
FUNCTION H(X,Y)=A*X+B*Y+C
A=.5;B=.7;C=1
FIT(A,B,C),H TO L
```

In EXAMPLE 3, we are trying to find the best planar fit to the data contained in the matrix L.

EXAMPLE MLAB 1

In this example, we fit a straight line F(X)=A*X+B to the input data M. After the FIT statement we press a carriage return as a response to the questions about constraints, number of iterations and convergence and hence we get the default values. Linear fits generally converge with ease within three iterations. Note that each iteration contains several sub-iterations.

```
*M COL 1=0:2:.2
*M COL 2=LIST(1.1,1.3,1.5,1.8,1.9,2.4,2.6,2.5,2.6,3.1,2.9)
*FUNCTION F(X)=A*X+B
*A=1;B=1
*FIT(A,B),F TO M

CONSTRAINTS NAME=(NONE)?
MAX NUMBER OF ITERATIONS=(3)?
CONVERGENCE FACTOR FOR SUM OF SQUARES =(.001)?

INITIAL PARAMETER VALUES:
             1.00000
             1.00000

BEGIN ITERATION 1
BEGIN ITERATION 1.1, SUM OF SQUARES= .510000
BEGIN ITERATION 1.2, SUM OF SQUARES= .245818
CURRENT PARAMETER VALUES:
          .981868
          1.17266

BEGIN ITERATION 2
BEGIN ITERATION 2.1, SUM OF SQUARES= .245818
BEGIN ITERATION 2.2, SUM OF SQUARES= .245818
BEGIN ITERATION 2.3, SUM OF SQUARES= .245818
BEGIN ITERATION 2.4, SUM OF SQUARES= .245818
BEGIN ITERATION 2.5, SUM OF SQUARES= .245818
BEGIN ITERATION 2.6, SUM OF SQUARES= .245818

FINAL                NORMAL ERROR        DEPENDENCY
PARAMETER VALUES:    STANDARD ERRORS:    VALUES:
     .981839           .787879@-1        .714286    A
     1.17270           .932231@-1        .714286    B

CONVERGED
RMS ERROR= .165267
FINAL SUM OF SQUARES= .245818
# OF ITERATIONS USED=2
```

Thus the best straight line fit for this data is given by

F(X)=(.981839)*X +(1.17270)

and the final sums of squares is .245818. The fitting process required only two iterations. In the following MLAB statements, we introduce the notation F on M. If X1,X2,..., XN are the elements in the first column of M, then in this case F on M produces the column vector with entries F(X1), F(X2),...,F(XN). We construct a three column matrix N which agrees with the data matrix M in the first two columns and which contains the values of the fitting function in the third column. The matrix N gives us a convenient way of displaying the original Y values and the fitted values side by side.

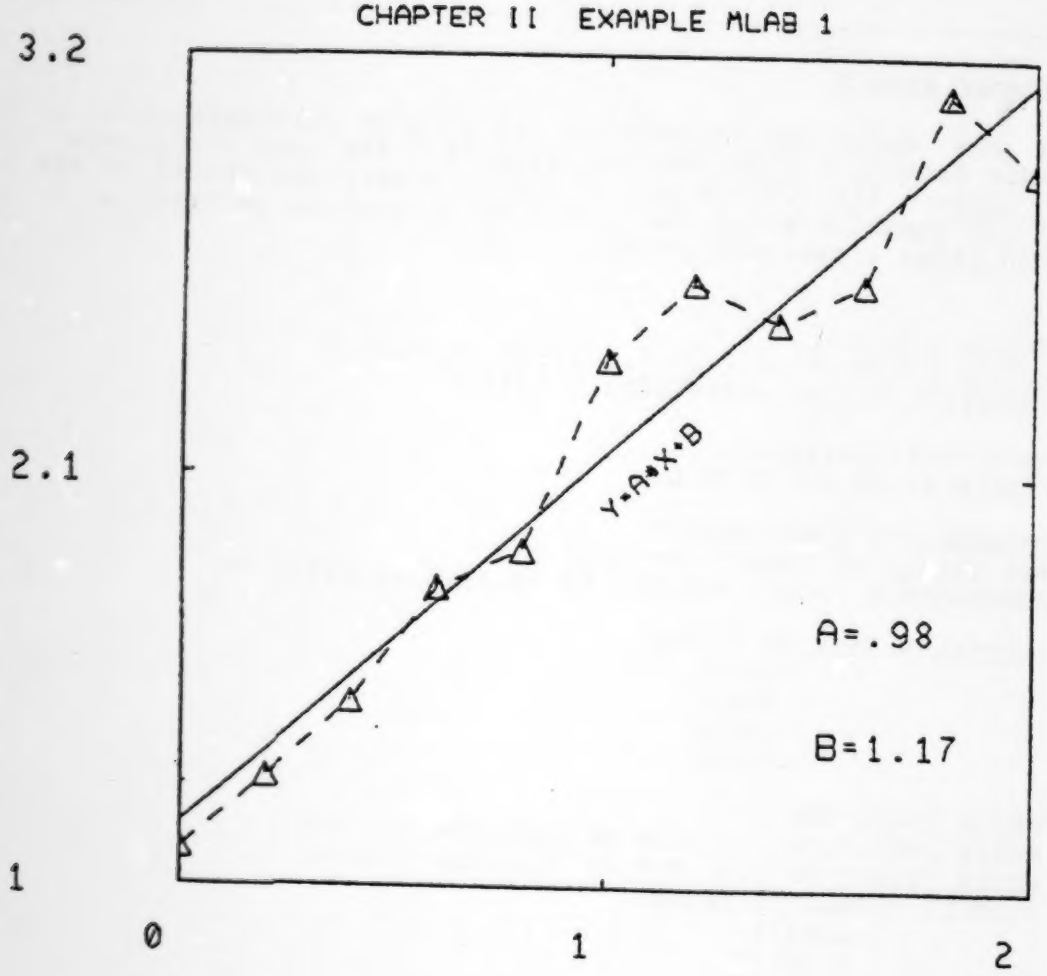
```
*N COL 1=M COL 1
*N COL 2=M COL 2
*N COL 3=F ON M
# ACTUALS NEQ # FORMALS FOR F
*TYPE N

MATRIX N:

 .0000000      1.100000      1.172703      *
 .2000000      1.300000      1.369070      *
 .4000000      1.500000      1.565438      *
 .6000000      1.800000      1.761806      *
 .8000000      1.900000      1.958174      *
1.000000       2.100000      2.154541      *
1.200000       2.600000      2.350909      *
1.400000       2.500000      2.547277      *
1.600000       2.600000      2.743645      *
1.800000       3.100000      2.940012      *
2.000000       2.900000      3.136380      *
```

To view the original data and the fitting curve graphically, we use the following commands

```
DRAW N COL(1,2),POINTTYPE 3,LINETYPE 2
DRAW N COL(1,3)
DISPLAY=T4012
```

The above commands result in a broken curve through the original data and a solid curve through the fitted values. The above commands yield axes labelled as follows: X-axis from 0 to 2, Y-axis from 1.1 to 3.136. This numbering of the axes comes from the DEFAULTWINDOW which scales the axes from the minimum X value to the maximum X and from the minimum Y value to the maximum Y.

We can change the numbering of either of these axes by introducing a curve whose graph is a horizontal line or a vertical line and which has a convenient maximum and minimum.


CHAPTER II EXAMPLE MLAB 1
3.2
2.1
1
0
1
2
Y=A*X+B
A=.98
B=1.17

In this case we adjust the Y-axis bY drawing a curve through (0,1), (0,1.2), ..., (0,3), (0,3.2).

```
*DRAW N COL(1,2),POINTTYPE 3,LINETYPE 2
*DRAW N COL(1,3)
*DRAW 0&'(1:3.2:.2)
*DISPLAY=T4012
*STRING A1="Y=A*X+B",AT 1,2,SIZE 3,ANGLE 45
*STRING B1="A=.98",AT 1.5,1.7,SIZE 4
*STRING C1="B=1.17",AT 1.5,1.4,SIZE 4
*DELETE DEFAULTWINDOW
```

EXAMPLE MLAB 2

In the following example we fit a cubic polynomial H(X) to the input data L. A cubic fit may require a few more iterations than a linear fit. So instead of typing a carriage return at the end of the question about the number of iterations we type a 5 and then press a carriage return.

```
*L COL 1=0:2.4:.3
*L COL 2=LIST(0,.8,1.1,.7,-.1,-.6,.4,.9,1.3)
*FUNCTION H(X)=A +A1*X+A2*X^2+A3*X^3

*A=0;A1=1;A2=1;A3=1
*FIT(A,A1,A2,A3),H TO L

CONSTRAINTS NAME=(NONE)?
MAX NUMBER OF ITERATIONS=(3)? 5
CONVERGENCE FACTOR FOR SUM OF SQUARES =(.001)?

INITIAL PARAMETER VALUES:
              .000000
              1.00000
              1.00000
              1.00000

BEGIN ITERATION 1
BEGIN ITERATION 1.1, SUM OF SQUARES= 841.453
BEGIN ITERATION 1.2, SUM OF SQUARES= 1.03990
CURRENT PARAMETER VALUES:
           .209187
           2.56169
          -3.41151
           1.07413

BEGIN ITERATION 2
BEGIN ITERATION 2.1, SUM OF SQUARES= 1.03990
BEGIN ITERATION 2.2, SUM OF SQUARES= 1.02462
CURRENT PARAMETER VALUES:
           .149937
           2.93586
```

```
          -3.80231
           1.17895

BEGIN ITERATION 3
BEGIN ITERATION 3.1, SUM OF SQUARES= 1.02462
BEGIN ITERATION 3.2, SUM OF SQUARES= 1.02438

FINAL                NORMAL ERROR        DEPENDENCY
PARAMETER VALUES:    STANDARD ERRORS:    VALUES:
     .142484             .419399          .870582     A
     2.98273             1.61339          .995713     A1
    -3.85138             1.62372          .998906     A2
     1.19214             .443954          .996810     A3

CONVERGED
RMS ERROR= .452633
FINAL SUM OF SQUARES= 1.02438
# OF ITERATIONS USED=3
```

Thus the best cubic fit is

$$H(X)=.142484+(2.98273)*X+(-3.85138)*X^2+(1.19214)*X^3$$

Below we construct a three column matrix L1 which contains in the first two columns the original data and the third column contains the fitted values.

```
*L1 COL 1=L COL 1
*L1 COL 2=L COL 2
*L1 COL 3=H ON L
# ACTUALS NEQ # FORMALS FOR H

*TYPE L1

MATRIX L1:

 .0000000      .0000000      .1424842      *
 .3000000      .8000000      .7228662      *
 .6000000     1.100000       .8031274      *
 .9000000      .7000000      .5763951      *
1.200000      -.1000000      .2357966      *
1.500000      -.6000000     -.2554071@-1   *
1.800000       .4000000     -.1448935@-1   *
2.100000       .9000000      .4620780      *
2.400000      1.300000      1.597289       *
```

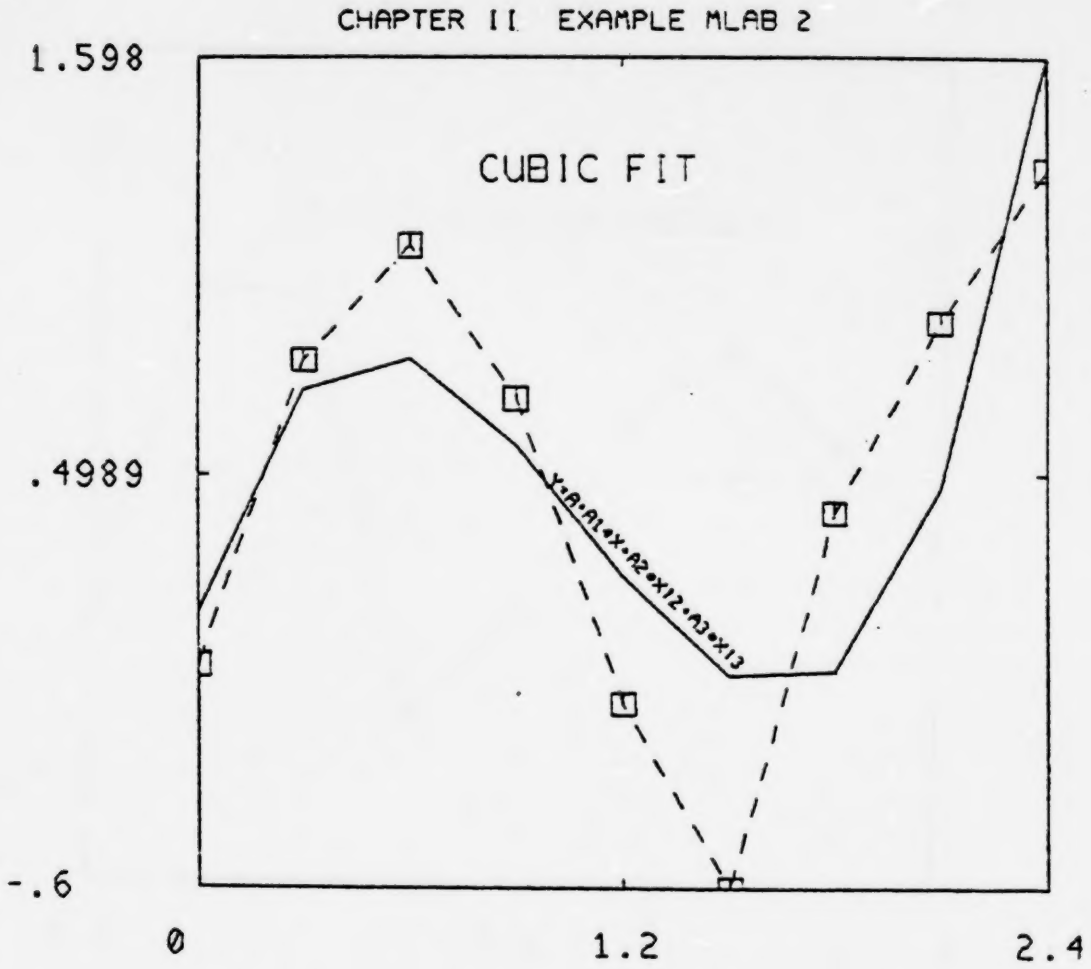
After looking at L1, we may decide that the fit is not as good as we had hoped and decide that perhaps with a more restrictive convergence criterion a better fit could be obtained. So we will go through the fit again, but this time in response to the convergence factor question we will type .00001 and then a carriage return. Note that we do not have to reinitialize the parameters because the parameters will carry as their new initial values the

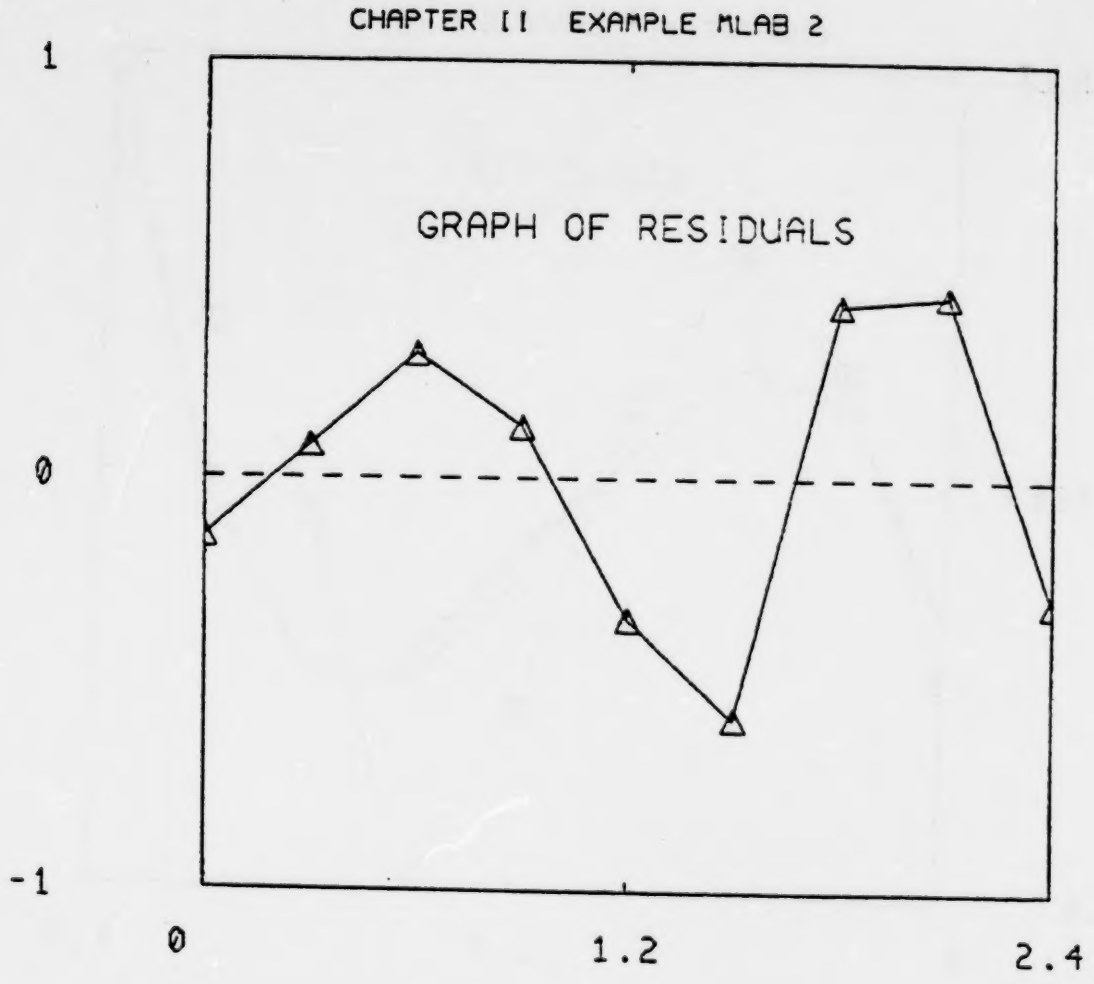
final values of the previous fit.

```
*FIT(A,A1,A2,A3),H TO L

CONSTRAINTS NAME=(NONE)?
MAX NUMBER OF ITERATIONS=(3)? 5
CONVERGENCE FACTOR FOR SUM OF SQUARES =(.001)?.00001

INITIAL PARAMETER VALUES:
              .142484
             2.98273
            -3.85138
             1.19214

BEGIN ITERATION 1
BEGIN ITERATION 1.1, SUM OF SQUARES= 1.02438
BEGIN ITERATION 1.2, SUM OF SQUARES= 1.02438

FINAL                  NORMAL ERROR          DEPENDENCY
PARAMETER VALUES:      STANDARD ERRORS:      VALUES:
     .141548             .419398            .870582     A
    2.98861             1.61339             .995713     A1
   -3.85754             1.62372             .998906     A2
    1.19380              .443953            .996810     A3


CONVERGED
RMS ERROR= .452632
FINAL SUM OF SQUARES= 1.02438
# OF ITERATIONS USED=1
```

The final sum of squares is the same for the two fits even though the parameters did change slightly. We now enter in the third column of L1 the new fitted values and compare them with the original data values in the second column of L1.

```
*L1 COL 3=H ON L
# ACTUALS NEQ # FORMALS FOR H
*TYPE L1

MATRIX L1:

 .0000000     .0000000     .1415485      *
 .3000000     .8000000     .7231863      *
 .6000000    1.100000      .8038627      *
 .9000000     .7000000     .5769733      *
1.200000     -.1000000     .2359139      *
1.500000     -.6000000    -.2591962@-1   *
1.800000      .4000000    -.1513135@-1   *
2.100000      .9000000     .4616741      *
2.400000     1.300000     1.597893       *
```


CHAPTER II EXAMPLE MLAB 2
1.598
CUBIC FIT
.4989
-.6
0
1.2
2.4
Y=A+A1*X+A2*X12+A3*X13


CHAPTER 11 EXAMPLE MLAB 2
GRAPH OF RESIDUALS
1
0
-1
0
1.2
2.4

```
*DRAW L1 COL(1,2).POINTTYPE 4,LINETYPE 2
*DRAW L1 COL(1,3)
*DISPLAY=T4012
*STRING C1="Y=A+A1*X+A2*X^2+A3*X^3",AT 1,.49,SIZE 2,ANGLE -45
*STRING C2="CUBIC FIT",AT .8,1.3,SIZE 4
*DELETE DEFAULTWINDOW
```

It is sometimes instructive to look at the differences between the original data values and the fitted values (the residuals) to see for example if there exists some pattern in the original data that is missed by the model. The residuals are contained in the second column of the matrix L2 below.

```
*L2 COL 1=L COL 1
*L2 COL 2=L1 COL 2 - L1 COL 3

*TYPE L2

MATRIX L2:

 .0000000    -.1415485     *
 .3000000     .7681365@-1  *
 .6000000     .2961373     *
 .9000000     .1230267     *
1.200000     -.3359139     *
1.500000     -.5740804     *
1.800000      .4151314     *
2.100000      .4383259     *
2.400000     -.2978928     *

*DRAW L2,POINTTYPE 3
*DRAW (0:2.4:.3)&'0,LINETYPE 2
*DRAW 0&'(-1:1:.2)
*STRING B1="GRAPH OF RESIDUALS",AT .6,.6,SIZE 4
```

In the graph of the residuals, a vertical straight line is drawn to adjust the scaling of the vertical axis.

EXAMPLE MLAB 3

In this example we fit an exponential function F(X)=A*EXP(-B*X) to the input data L. A brief examination of the data shows that F(0)=A should be positive and that since the data exhibits a decreasing trend B should also be positive. Consequently we enter the constraint (named D1) in the fitting routine that A and B should be positive. After entering the data matrix L and the function F into MLAB we enter the statement

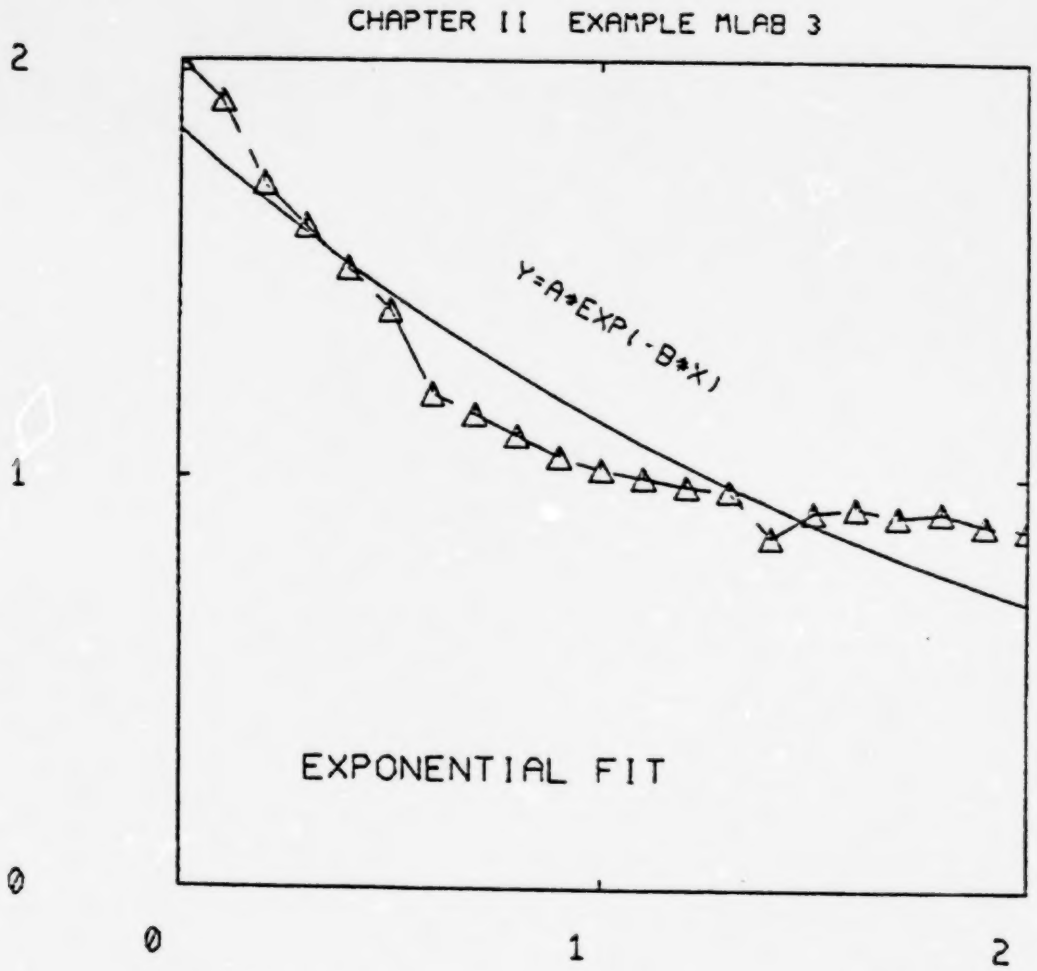
and press a carriage return. MLAB will then prompt the user for a reply with ! and then we type A>0 and press a carriage return. After the second ! we type B>0 and then a carriage return. After the third ! we simply press a carriage return.

In this example we use the MLAB command QUIET FIT rather than FIT; the result is the suppression of the results of the intermediate iterations - only the final results are given. We answer D1 followed by a carriage return to the MLAB question concerning the constraints name and we allow for 10 iterations since exponential and logarithmic fits generally require more iterations than are required by a linear fit.

```
*L COL 1=0:2:.1
*L COL 2=LIST(2,1.9,1.7,1.6,1.5,1.4,1.2,1.15,1.1,1.05,1.02,1,.98,
.97,.86,.92,.93,.91,.92,.89,.88)
*FUNCTION F(X)=A*EXP(-B*X)
*CONSTRAINTS D1
!A>0
!B>0
!
*A=1;B=1
*QUIET FIT(A,B),F TO L

CONSTRAINTS NAME=(NONE)? D1
MAX NUMBER OF ITERATIONS=(3)? 10
CONVERGENCE FACTOR FOR SUM OF SQUARES =(.001)?


FINAL                NORMAL ERROR        DEPENDENCY
PARAMETER VALUES:    STANDARD ERRORS:    VALUES:
     1.82865              .614193@-1       .598427     A
     .479100              .387375@-1       .598427     B
LAGRANGE MULTIPLIERS:
          .000000
          .000000
CONVERGED
RMS ERROR= .120144
FINAL SUM OF SQUARES= .274255
# OF ITERATIONS USED=4
```

Note that the input for the second column of the matrix L required two lines. In this case a carriage return was not typed at the end of the first line; we simply continued entering data until the typewriter automatically shifted to the next line. The result is that MLAB considers the two typewritten lines as a single line of input.

The fitting routine converged in 4 iterations giving as a best exponential fit


CHAPTER II EXAMPLE MLAB 3
2
1
0
0
1
2
Y=A*EXP(-B*X)
EXPONENTIAL FIT

```
F(X)=(1.82865)*EXP(-.479100*X)
```

As in the previous examples we type out a matrix which displays the data and the fitted values side by side.

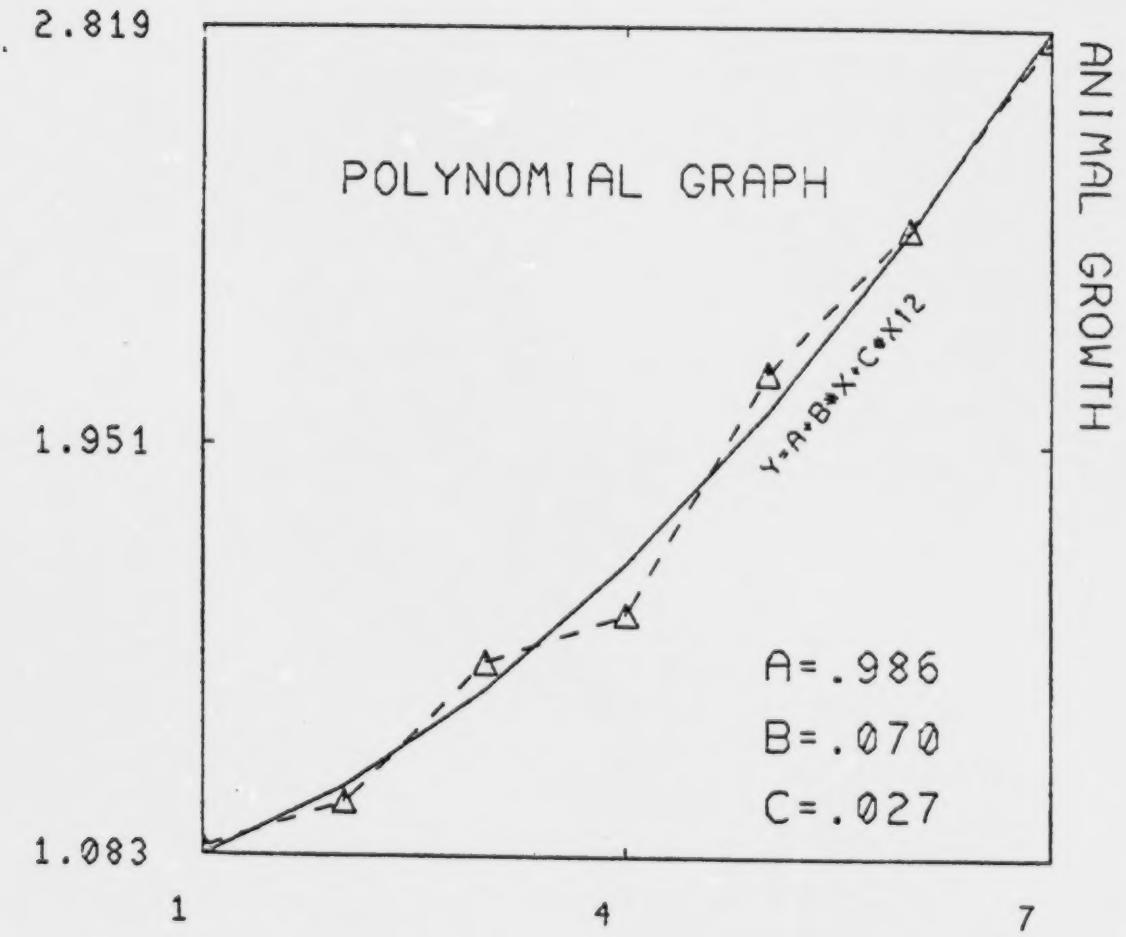
```
*L1 COL 1=L COL 1
*L1 COL 2=L COL 2
*L1 COL 3=F ON L
# ACTUALS NEQ # FORMALS FOR F
*TYPE L1

MATRIX L1:

 .0000000      2.000000      1.828647      *
 .1000000      1.900000      1.743102      *
 .2000000      1.700000      1.661559      *
 .3000000      1.600000      1.583831      *
 .4000000      1.500000      1.509738      *
 .5000000      1.400000      1.439112      *
 .6000000      1.200000      1.371790      *
 .7000000      1.150000      1.307617      *
 .8000000      1.100000      1.246446      *
 .9000000      1.050000      1.188137      *
1.000000       1.020000      1.132555      *
1.100000       1.000000      1.079574      *
1.200000       .9800000      1.029071      *
1.300000       .9700000      .9809302      *
1.400000       .8600000      .9350418      *
1.500000       .9200000      .8913001      *
1.600000       .9300000      .8496047      *
1.700000       .9100000      .8098598      *
1.800000       .9200000      .7719742      *
1.900000       .8900000      .7358609      *
2.000000       .8800000      .7014370      *


*DRAW L1 COL(1,2),POINTTYPE 3,LINETYPE 4
*DRAW L2 COL(1,3)
*DRAW 0&'(0:2:.2)
*DISPLAY=T4012
*STRING C1="EXPONENTIAL FIT",AT .3,.3,SIZE 4
*STRING C2="Y=A*EXP(-B*X)",AT .8,1.5,SIZE 3,ANGLE -30
*DELETE DEFAULTWINDOW
```

EXAMPLE MLAB 4

In this example we fit a quadratic polynomial to a matrix M of input data and then we graphically display the original data versus the fitting curve.

```
*M COL 1=1:7
*M COL 2=LIST(1.1,1.2,1.5,1.6,2.1,2.4,2.8)

*FUNCTION F(X)=A+B*X+C*X^2
*A=1;B=1;C=1

*QUIET FIT(A,B,C),F TO M;!;!10;!;


FINAL                NORMAL ERROR         DEPENDENCY
PARAMETER VALUES:    STANDARD ERRORS:     VALUES:
     .985731             .116569          .941177     A
     .702279@-1          .668047@-1       .991045     B
     .273822@-1          .816150@-2       .982036     C

CONVERGED
RMS ERROR= .748013@-1
FINAL SUM OF SQUARES= .223809@-1
# OF ITERATIONS USED=3

*DRAW M, POINTTYPE 3,LINETYPE 2

*DRAW M COL 1&'(F ON M)

# ACTUALS NEQ # FORMALS FOR F

*STRING C1="Y=A+B*X+C*X^2",AT 5,1.9,SIZE 3,ANGLE 50

*STRING C2="POLYNOMIAL GRAPH",AT 2,2.5,SIZE 5

*STRING C3="ANIMAL GROWTH",AT 7.3,2.8,ANGLE -90

*STRING C4="A=.986",AT 5,1.5

*STRING C5="B=.070",AT 5,1.35

*STRING C6="C=.027",AT 5,1.2

*DISPLAY=T4012

*DELETE DEFAULTWINDOW

*EXIT
```

The ! in a FIT statement allows the user to answer the three questions concerning the FIT in one line of commands. This is particularly useful in DO FILES.


2.819
1.951
1.083
1
4
7
POLYNOMIAL GRAPH
ANIMAL GROWTH
Y=A+B*X+C*X12
A=.986
B=.070
C=.027

CHAPTER III: DIFFERENTIAL EQUATIONS

In the first section of this chapter we discuss the process of taking derivatives and integrals in closed form. In the remaining sections we treat the numerical solution of first order equations and sytems of first order equations.

III.1 THE DIFF AND INTEGRAL OPERATORS

If a function F(X) has been defined in MLAB then the function F DIFF X will be the derivative of F with respect to X. Similarly if a function G(X,Y) has been defined in MLAB then G DIFF X and G DIFF Y will be then partial derivatives of G with respect to X and Y respectively. The same process applies to a function H(X1,X2,...,XN) of N variables.

EXAMPLE 1.

```
FUNCTION F(X)=X^3-SIN(X)
TYPE F DIFF X
TYPE F DIFF X(2)
```

The result of EXAMPLE 1 will be the derivative of F printed out followed by the derivative of F evaluated at X=2.

EXAMPLE 2.

```
FUNCTION G(X,Y)=X*Y + EXP(X-Y)
TYPE G DIFF X
TYPE G DIFF Y
TYPE G DIFF X(2,-3)
TYPE G DIFF Y(0,4)
```

The partials and the partials evaluated at the indicated points will be printed out from the instructions in EXAMPLE 2.

The DIFF operator interacts in the usual way with sums, products, quotients and the composition of functions.

EXAMPLE 3 .

```
FUNCTION K(X)=(X^3+X+1)/COS(1+EXP(X))
TYPE K DIFF X
TYPE K DIFF X(0)
```

In EXAMPLE 3 the MLAB operator will apply the quotient rule and the chain rule to obtain the appropriate derivative. The command TYPE K DIFF X(X) is not appropriate; the function is K DIFF X, whereas K DIFF X(B) is a scalar and hence the argument B must be a number and not a general variable.

The MLAB command INTEGRAL permits the user to compute the definite integral of a function given in closed form.

EXAMPLE 4.

```
FUNCTION F(X)=X-3
FUNCTION G(A,B)=INTEGRAL(X,A,B,F(X))
TYPE G(-1,4)
```

G(A,B) equals the definite integral of the function F(X) from the lower limit of integration A to the upper limit of integration B. Note that the INTEGRAL operator is used to define the function G; the INTEGRAL operator must be used in this way, that is, in the definition of a function. The command

```
TYPE INTEGRAL(X,-1,4,F(X))
```

is an illegal command in MLAB.

EXAMPLE 5.

```
FUNCTION K(T)=T^2 - LOG(2*T+3)
FUNCTION L(C,D)=INTEGRAL(T,C,D,K(T))
TYPE L(2,4)
```

As a result of the instructions in EXAMPLE 5, the value of the definite integral of K(T) from 2 to 4 will be printed out.

The INTEGRAL operator must have both the lower and upper limits of integration as arguments (A and B in EXAMPLE 4, C and D in EXAMPLE 5). By fixing either the upper or the lower limit the INTEGRAL operator can be used to define a function of one variable rather than two.

EXAMPLE 6.

```
FUNCTION R(Y)=(SIN(2*Y)^3)
FUNCTION S(T)=INTEGRAL(Y,0,T,R(Y))
M=POINTS(S,1:5)
TYPE M
```

Notice that the first argument in the INTEGRAL operator is the variable of integration, that is, it must agree with the argument of the integrand which appears as the last argument of the INTEGRAL operator.

In EXAMPLE 6 we fixed the lower limit of integration and consequently S is a function of the single variable T. The matrix M contains in its second column the values of the definite integral with upper limits of integration equal to 1,2,...,5, respectively.

EXAMPLE MLAB 1

```
*FUNCTION F(X)=X^2-3*X+7
*TYPE F DIFF X
```

```
FUNCTION F DIFF X(X) = 2*X-3

*TYPE F DIFF X(2)

SCALAR = 1.000000
```

EXAMPLE MLAB 2

```
*FCT G(X)=X*SIN(X)
*TYPE G DIFF X

FUNCTION G DIFF X(X) = X*COS(X)+SIN(X)
```

In this example MLAB applied the product rule to obtain the desired derivative.

EXAMPLE MLAB 3

```
*FUNCTION H(X)=G(X)/F(X)
*TYPE H DIFF X

FUNCTION  H DIFF X(X) = G DIFF X(X)/F(X)+G(X)*(-F(X)^(-2))*F DIFF
X(X)
```

In this example F(X) and G(X) are as defined in EXAMPLES MLAB 1 and 2, but MLAB ignored the particular values of F and G and just computed the quotient rule in general.

EXAMPLE MLAB 4

```
*FUNCTION L(X)=(X+1)/SIN(X)
*TYPE L DIFF X

FUNCTION L DIFF X(X) = 1/SIN(X)+(X+1)*(-SIN(X)^(-2))*COS(X)
```

In this example the numerator and the denominator are explicitly represented in L(X) and hence the derivative is represented in terms of these functions.

EXAMPLE MLAB 5

```
*FUNCTION F1(X)=SIN(X^2+1)
*TYPE F1 DIFF X

FUNCTION F1 DIFF X(X) = COS(X^2+1)*2*X
```

In this example MLAB applied the chain rule to obtain the appropriate result.

EXAMPLE MLAB 6

```
*FUNCTION H1(X)=INTEGRAL(T,0,X,EXP(T))
*TYPE H1(1)

SCALAR = 1.718283
*TYPE EXP(1)

SCALAR = 2.718282
```

The exact value of the definite integral, H1(1), is EXP(1)-1 and hence we can compare H1(1) and EXP(1)-1 to get an idea of the accuracy of the integration routine for the given integrand.

EXAMPLE MLAB 7

```
*FUNCTION L1(X)=1/(1+X)
*FUNCTION L2(A,B)=INTEGRAL(X,A,B,L1(X))
*TYPE L2(0,1)

SCALAR = .6931489
```

Since the integral of 1/(1+X) is equal to LOG(1+X), L2(0,1)=LOG(2)-LOG(1)=LOG(2); hence we can compare the answer above with LOG(2)=.6931472 .

EXAMPLE MLAB 8

```
*FUNCTION P(X)=1+X+X^2

*FUNCTION P1(X)=INTEGRAL(T,0,X,P(T))
*TYPE P1 DIFF X

FUNCTION P1 DIFF X(X) = P(X)

*TYPE P1(0)

SCALAR = .0000000
```

In this example, just as a check, we verified that in MLAB if P1(X) is the integral of P(X), then P1'(X)=P(X)

Note that P1(X)=INTEGRAL(T,0,X,P(T)) yields a solution to the differential equation DY/DX = P(X)

with the initial condition Y(Ø)=Ø.

III.2 FIRST ORDER DIFFERENTIAL EQUATIONS

In this section we look at first order differential equations of the form

$$Y' = F(X,Y)$$

with initial condition Y(A)=B. For the solution, MLAB uses the operator INTEGRATE. The function F(X,Y), the constants A and B and the X values for which the solution is desired are the input required for the INTEGRATE operator.

EXAMPLE 1.

```
FCT Y DIFF X(X)=-Y*X^2
INITIAL Y(1)=3
M=INTEGRATE(Y DIFF X,1:5:.2)
```

The solution matrix M will contain three columns. The first column contains those X values specified in the second argument or the INTEGRATE operator which in this case are the numbers 1,1.2,1.4,...,4.6,4.8,5.0. The second column of M contains the values of the solution Y(X) and the third column the Y'(X) values.

EXAMPLE 2.

```
FUNCTION S DIFF T(T)=SIN(T)/(S(T)^2)
INITIAL S(2)=1
N=INTEGRATE(S DIFF T,2:3:.1)
DRAW N COL(1,2)
DISPLAY=T4010
```

Note that the first T value at which the solution is desired must agree with the T value at which the initial condition is given. The DRAW command above will cause the solution curve to be drawn on the display.

EXAMPLE 3.

```
FCT P DIFF Z(Z)=P(Z)^2*Z^3-P(Z)*Z
INITIAL P(0)=2
L=INTEGRATE(P DIFF Z,0:10:.4)
TYPE L
```

The TYPE L command will cause the three column matrix L to be typed out. The first column will contain the Z values 0,.4,.8,...,9.6,10.0 and the second column will contain the corresponding values of P:P(0),P(.4),...,P(9.6),P(10.0).

EXAMPLE MLAB 1

```
*FUNCTION Y DIFF X(X)=(X^2+1)/(X*Y)
*INITIAL Y(1)=3
*M=INTEGRATE(Y DIFF X,1:7:.3)
*TYPE M
```

MATRIX M:

```
 1.000000      3.000000      .6666667      *
 1.300000      3.196057      .6474504      *
 1.600000      3.391166      .6562013      *
 1.900000      3.590806      .6757202      *
 2.200000      3.796992      .6991222      *
 2.500000      4.010339      .7231372      *
 2.800000      4.230779      .7462331      *
 3.100000      4.457921      .7677538      *
 3.400000      4.691248      .7874494      *
 3.700000      4.930206      .8052950      *
 4.000000      5.174249      .8213753      *
 4.300000      5.422868      .8358230      *
 4.600000      5.675594      .8487904      *
 4.900000      5.932009      .8604302      *
 5.200000      6.191735      .8708878      *
 5.500000      6.454436      .8802962      *
 5.800000      6.719819      .8887756      *
 6.100000      6.987620      .8964329      *
 6.400000      7.257610      .9033627      *
 6.700000      7.529576      .9096471      *
 7.000000      7.803338      .9153592      *
```

The exact solution to the above differential equation is given by the function S below. We type out below the first two columns of the solution matrix M and then we adjoin (via the operator &´) a third column which contains the values of the exact solution to the differential equation and hence we can compare the solution obtained by the INTEGRATE operator with the exact solution.

```
*FUNCTION S(X)=SQRT(X^2+2*LOG(X)+8)
*S1=POINTS(S,1:7:.3)
*TYPE M COL(1,2)&´(S1 COL 2)
```

MATRIX :

```
 1.000000      3.000000      3.000000      *
 1.300000      3.196057      3.196049      *
 1.600000      3.391166      3.391166      *
 1.900000      3.590806      3.590781      *
 2.200000      3.796992      3.796961      *
 2.500000      4.010339      4.010309      *
 2.800000      4.230779      4.230749      *
 3.100000      4.457921      4.457892      *
 3.400000      4.691248      4.691221      *
 3.700000      4.930206      4.930179      *
```

```
 4.000000      5.174249      5.174224      *
 4.300000      5.422868      5.422844      *
 4.600000      5.675594      5.675572      *
 4.900000      5.932009      5.931987      *
 5.200000      6.191735      6.191714      *
 5.500000      6.454436      6.454417      *
 5.800000      6.719819      6.719800      *
 6.100000      6.987620      6.987602      *
 6.400000      7.257610      7.257589      *
 6.700000      7.529576      7.529556      *
 7.000000      7.803338      7.803321      *
```

```
EXAMPLE MLAB 2

FUNCTION Y DIFF X(X) = Y*SIN(X)
*INITIAL Y(0)=2
*PI=3.14159
*M=INTEGRATE(Y DIFF X,0:PI:.2)
*TYPE M

MATRIX M:

 .0000000      2.000000      .0000000      *
 .2000000      2.040276      .4053389      *
 .4000000      2.164284      .8428083      *
 .6000000      2.381687      1.344800      *
 .8000000      2.708611      1.943032      *
1.000000       3.167167      2.665053      *
1.200000       3.783986      3.526767      *
1.400000       4.586717      4.519900      *
1.600000       5.597559      5.595108      *
1.800000       6.823284      6.644835      *
2.000000       8.242367      7.494773      *
2.200000       9.792774      7.917520      *
2.400000       11.36506      7.676934      *
2.600000       12.80771      6.602756      *
2.800000       13.94873      4.672935      *
3.000000       14.63122      2.064759      *
```

The exact solution is given by the function K(X) defined below and in the matrix L1 we place the numerical solution from MLAB in the first column and the values of the exact solution in the second column for purposes of comparison.

```
*A=2*EXP(1)
*FUNCTION K(X)=A*EXP(-COS(X))
*L=POINTS(K,0:PI:.2)
*
*L1 COL 1=M COL 2
*L1 COL 2=L COL 2
*TYPE L1
```

MATRIX L1:

2.000000	2.000000	*
2.040276	2.040267	*
2.164284	2.164277	*
2.381687	2.381693	*
2.708611	2.708623	*
3.167167	3.167190	*
3.783986	3.784029	*
4.586717	4.586788	*
5.597559	5.597649	*
6.823284	6.823341	*
8.242367	8.242420	*
9.792774	9.792808	*
11.36506	11.36503	*
12.80771	12.80756	*
13.94873	13.94847	*
14.63122	14.63096	*

III.3 SECOND ORDER DIFFERENTIAL EQUATIONS

MLAB is set up to solve first order differential equations and systems of first order differential equations, so to to solve a given differential equation or system of equations in MLAB we must recast the equation or system of equations into one of these two forms.

EXAMPLE 1.

In this example we wish to solve the second order differential equation

$$Y'' + Y = 0$$

with the initial conditions

$$Y(0)=0 \text{ and } Y'(0)=1$$

to solve this equation we introduce the new variable Z=dY/dX and recast the original second order equation into a system of first order equations in dZ/dX and dY/dX. We will ask MLAB for the solution at the points 0,.3,.6,...,2.7,3.0.

```
FUNCTION Y DIFF X(X)=Z(X)
FUNCTION Z DIFF X(X)=-Y(X)
INITIAL Y(0)=0
INITIAL Z(0)=1
K=INTEGRATE(Y DIFF X,Z DIFF X,0:3:.3)
TYPE K
DRAW K COL(1,2)
DISPLAY=T4010
```

The matrix K will have 5 columns. The first column will contain the X values which are specified in the last argument of the INTEGRATE operator; in this case, the first column contains the numbers 0 to 3 in increments of .3. The second column contains the solution values Y(X) corresponding to the X values in the first column. The remaining columns contain the values Y'(X), Z(X), and Z'(X). Note that in this case the third and fourth columns contain the same values.

The DRAW command will cause a solid curve to be drawn through the solution points (X,Y(X)).

EXAMPLE 2.

In this example we wish to solve the second order differential equation

$$Y'' + Y' + Y = 0$$

with the initial conditions

Y(1)=1 and Y′(1)=2

To solve this equation we introduce the variable Z=dY/dX and set up equations in dZ/dX and dY/dX just as we did in EXAMPLE 1.

```
FUNCTION Y DIFF X(X)=Z(X)
FUNCTION Z DIFF X(X)=-Z(X)-Y(X)
INITIAL Y(1)=1
INITIAL Z(1)=2
N=INTEGRATE(Y DIFF X,Z DIFF X,1:3:.2)
TYPE N
```

The result of the above instructions will be a 5 column matrix containing, respectively, the values X, Y(X), Y′(X), Z(X) and Z′(X).

EXAMPLE 3.

In this example we wish to solve the second order differential equation

Y′′ + 3*Y′ - 2*Y = SIN(X)

with the initial conditions

Y(0)=1 and Y′(0)=0

```
FUNCTION Y DIFF X(X)=Z(X)
FUNCTION Z DIFF X(X)=-3*Z(X)+2*Y(X)+SIN(X)
INITIAL Y(0)=1
INITIAL Z(0)=0
M=INTEGRATE(Y DIFF X,Z DIFF X,0:2:.2)
```

As in EXAMPLES 1 and 2, we set up equations in dY/dX and dZ/dX and the output matrix M has 5 columns. The second column contains the solution values Y(0),Y(.2),...,Y(2.0).

EXAMPLE MLAB 1

In this example we will solve the second order equation

Y′′ - Y′ + 2*Y = 0 with the initial conditions

Y(0)=1 and Y′(0)=2

The exact general solution of the above homogeneous equation is H(X)=C1*EXP(2*X)+C2*EXP(-X) and from the initial conditions we get the values C1=1 and C2=0. Below we compare the MLAB solution with the exact solution.

```
*FUNCTION Y DIFF X(X)=Z(X)
*FUNCTION Z DIFF X(X)=Z(X)+2*Y(X)
*INITIAL Y(0)=1
*INITIAL Z(0)=2
*K=INTEGRATE(Y DIFF X,Z DIFF X,0:1:.1)
*TYPE K

MATRIX K:

 .0000000     1.000000     2.000000     2.000000     4.000000     *
 .1000000     1.221407     2.442814     2.442814     4.885628     *
 .2000000     1.491829     2.983658     2.983658     5.967316     *
 .3000000     1.822124     3.644247     3.644247     7.288495     *
 .4000000     2.225547     4.451094     4.451094     8.902187     *
 .5000000     2.718288     5.436576     5.436576     10.87315     *
 .6000000     3.320122     6.640244     6.640244     13.28049     *
 .7000000     4.055206     8.110411     8.110411     16.22082     *
 .8000000     4.953039     9.906078     9.906078     19.81216     *
 .9000000     6.049655     12.09931     12.09931     24.19862     *
1.000000      7.389058     14.77812     14.77812     29.55623     *

*FUNCTION H(X)=EXP(2*X)
*R=POINTS(H,0:1:.1)
*TYPE K COL(1,2)&'(R COL 2)

MATRIX :

 .0000000     1.000000     1.000000     *
 .1000000     1.221407     1.221403     *
 .2000000     1.491829     1.491825     *
 .3000000     1.822124     1.822119     *
 .4000000     2.225547     2.225541     *
 .5000000     2.718288     2.718282     *
 .6000000     3.320122     3.320117     *
 .7000000     4.055206     4.055200     *
 .8000000     4.953039     4.953032     *
 .9000000     6.049655     6.049648     *
1.000000      7.389058     7.389056     *
```

EXAMPLE MLAB 2

In this example we will solve the differential equation

$$Y'' + Y' = 0$$

with the initial conditions

$$Y(0)=2 \text{ AND } Y'(0)=1$$

We use the MLAB INTEGRATE operator to solve the equation and then we compare the MLAB solution to the exact solution G(X)=3-EXP(-X).

```
*FUNCTION Y DIFF X(X)=Z(X)
*FUNCTION Z DIFF X(X)=-Z(X)
*INITIAL Y(0)=2
*INITIAL Z(0)=1
*M=INTEGRATE(Y DIFF X,Z DIFF X,0:2:.2)
*TYPE M

MATRIX M:

 .0000000     2.000000     1.000000     1.000000    -1.000000     *
 .2000000     2.181271      .8187289     .8187289    -.8187289     *
 .4000000     2.329682      .6703184     .6703184    -.6703184     *
 .6000000     2.451190      .5488106     .5488106    -.5488106     *
 .8000000     2.550672      .4493284     .4493284    -.4493284     *
1.000000      2.632121      .3678791     .3678791    -.3678791     *
1.200000      2.698806      .3011938     .3011938    -.3011938     *
1.400000      2.753403      .2465968     .2465968    -.2465968     *
1.600000      2.798103      .2018965     .2018965    -.2018965     *
1.800000      2.834702      .1652990     .1652990    -.1652990     *
2.000000      2.864665      .1353353     .1353353    -.1353353     *

*FUNCTION G(X)=3-EXP(-X)
*L=POINTS(G,0:2:.2)
*TYPE M COL(1,2)&'(L COL 2)

MATRIX :

 .0000000     2.000000     2.000000     *
 .2000000     2.181271     2.181269     *
 .4000000     2.329682     2.329680     *
 .6000000     2.451190     2.451188     *
 .8000000     2.550672     2.550671     *
1.000000      2.632121     2.632121     *
1.200000      2.698806     2.698806     *
1.400000      2.753403     2.753403     *
1.600000      2.798103     2.798104     *
1.800000      2.834702     2.834701     *
2.000000      2.864665     2.864665     *
```

EXAMPLE MLAB 3

This example exhibits the capability of MLAB to deal with a system of differential equations. MLAB can deal explicitly only with differential equations of the first order; the following system of equations is precisely in the form of two first order equations and hence can be solved by MLAB in its present form. If the system were not in this form, it would have to be recast in the form of a single first order equation or as a system of first order equations.

Also, since the solution of a differential equation in MLAB is an iterative process an appropriate starting point (an initial condition) for the iteration must be given.

```
*FUNCTION B DIFF X(X)=-(K1+K3)*B(X)+R
*FUNCTION D DIFF X(X)=K1*B(X)-K2*D(X)-A
*K1=.26;K2=.11;K3=.36; A=.43; R=2.8
*INITIAL B(0)=5.2
*INITIAL D(0)=2.8
*M=INTEGRATE(D DIFF X, B DIFF X,0:50:2)
```

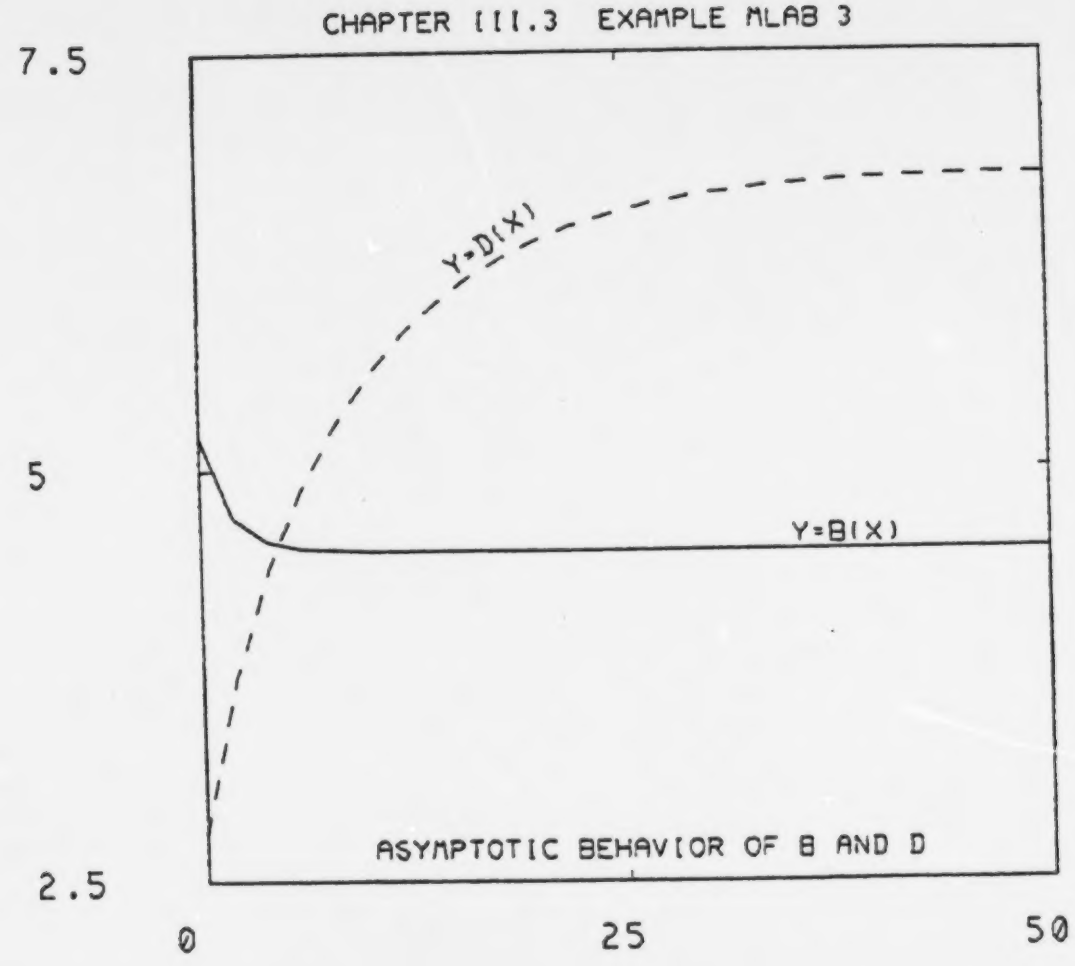
The matrix M will have five columns:

X	D(X)	D'(X)	B(X)	B'(X)

```
*TYPE M

MATRIX M:

 .0000000     2.800000     .6140000     5.200000    -.4240000     *
2.000000      3.760908     .3825077     4.716183    -.1240334     *
4.000000      4.405406     .2752394     4.576285    -.3729654@-1  *
6.000000      4.886692     .2114307     4.534488    -.1138233@-1  *
8.000000      5.261844     .1671011     4.522707    -.4078478@-2  *
10.00000      5.560075     .1332686     4.518757    -.1629248@-2  *
12.00000      5.798690     .1065675     4.517013    -.5479343@-3  *
14.00000      5.989804     .8539513@-1  4.516437    -.1907308@-3  *
16.00000      6.143009     .6849477@-1  4.516253    -.7670383@-4  *
18.00000      6.265928     .5495266@-1  4.516172    -.2663282@-4  *
20.00000      6.364576     .4409402@-1  4.516144    -.9019985@-5  *
22.00000      6.443721     .3538578@-1  4.516135    -.3575116@-5  *
24.00000      6.507237     .2839802@-1  4.516131    -.1284198@-5  *
26.00000      6.558221     .2278950@-1  4.516130    -.4574879@-6  *
28.00000      6.599127     .1828965@-1  4.516129    -.1706086@-6  *
30.00000      6.631957     .1467829@-1  4.516129    -.8484978@-7  *
32.00000      6.658310     .1177952@-1  4.516129    -.5883889@-7  *
34.00000      6.679452     .9453658@-2  4.516128    -.1897805@-7  *
36.00000      6.696425     .7587017@-2  4.516130     .5246714@-8  *
38.00000      6.710043     .6088664@-2  4.516128    -.1279510@-9  *
40.00000      6.720972     .4886481@-2  4.516128     .0000000     *
42.00000      6.729746     .3921637@-2  4.516130     .0000000     *
44.00000      6.736784     .3147157@-2  4.516128     .0000000     *
46.00000      6.742433     .2525775@-2  4.516128     .0000000     *
48.00000      6.746970     .2027059@-2  4.516130     .0000000     *
50.00000      6.750607     .1626730@-2  4.516129     .0000000     *
```

We now draw the graphs of B(X) and D(X) and use different LINE-TYPES to aid in the comparison of the two curves.

```
*DRAW M COL(1,2),LINETYPE 2
*DRAW M COL(1,4)
*DISPLAY=T4012
*DRAW 0&'(2.5:7.5:.5)
*STRING A1="Y=D(X)",AT 15,6.2,SIZE 3,ANGLE 35
*STRING A2="Y=B(X)",AT 35,4.6,SIZE 3
*STRING A3="ASYMPTOTIC BEHAVIOR OF B AND D",AT 10,2.7,SIZE 3
*DELETE DEFAULTWINDOW
```


CHAPTER III.3 EXAMPLE MLAB 3
7.5
5
2.5
0
25
50
Y=D(X)
Y=B(X)
ASYMPTOTIC BEHAVIOR OF B AND D

CHAPTER IV: SOME SPECIAL MLAB COMMANDS

In this chapter we discuss and give examples of four MLAB commands. These and other commands are listed in the appendix.

IV.1 FUNCTION DEFINITIONS

We have already seen in Chapter 1 how to define a function if it is given by a single formula or expression. In this section we will discuss the way in which a function can be defined if it is composed of pieces of other functions.

EXAMPLE 1.

```
FUNCTION F(X)=IF X<2 THEN X^2 ELSE 1-X
```

The function F(X) defined in EXAMPLE 1 is equal to X^2 if X is less than 2 otherwise it is equal to 1-X. So, for example F(-1)=1 and F(3)=-2.

EXAMPLE 2.

```
FUNCTION G(X)=IF X<=1 THEN SIN(X) ELSE SIN(2*X)
```

The function G(X) defined in EXAMPLE 2 is equal to SIN(X) if X is less than or equal to 1 otherwise it equals SIN(2*X).

EXAMPLE 3.

```
FUNCTION K(X)= IF X>1 AND X<2 THEN X^2 ELSE 1/(1+X^2)
```

The function K(X) in EXAMPLE 3 is X^2 for X strictly between 1 and 2 and equals 1/(1+X^2) elsewhere.

EXAMPLE 4.

```
FUNCTION F(X)=X^2*EXP(-X)
FUNCTION G(X)=2-X
FUNCTION H(X)=X^2-4
FUNCTION K(X)=IF X<1 THEN F(X) ELSE IF X<2 THEN G(X) ELSE
H(X)
```

In EXAMPLE 4, K(X) is defined as follows: the first 'IF' statement is encountered. If it is true the 'THEN' statement is processed; if it is false the definition jumps over the 'THEN' statement to the 'ELSE' statement. The second 'IF' statement is encountered and is treated in the same way as the first 'IF' statement. The function K(X) in this example is equivalent to the following

```
                    F(X)    IF X<1
K(X)=               G(X)    IF 1<=X<2
                    H(X)    IF X>2
```

EXAMPLE MLAB 1

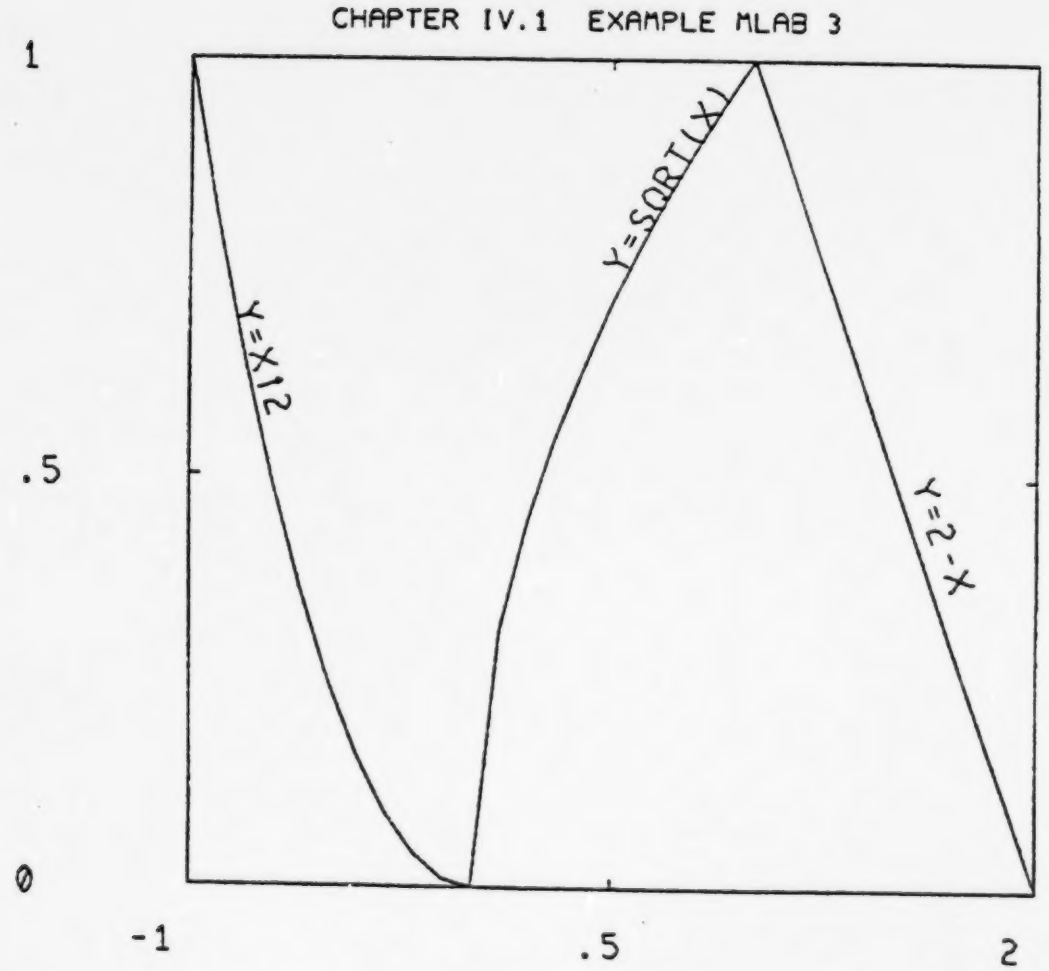
```
*FUNCTION R(X)=IF X<1 THEN 2 ELSE 4
*TYPE POINTS(R,-5:5)

MATRIX :

-5.000000      2.000000      *
-4.000000      2.000000      *
-3.000000      2.000000      *
-2.000000      2.000000      *
-1.000000      2.000000      *
 .0000000      2.000000      *
 1.000000      4.000000      *
 2.000000      4.000000      *
 3.000000      4.000000      *
 4.000000      4.000000      *
 5.000000      4.000000      *
```

EXAMPLE MLAB 2

```
*FUNCTION S(X)=IF X<-1 THEN 2 ELSE IF X<2 THEN 4 ELSE 6
*TYPE POINTS(S,-5:5)

MATRIX :

-5.000000      2.000000      *
-4.000000      2.000000      *
-3.000000      2.000000      *
-2.000000      2.000000      *
-1.000000      4.000000      *
 .0000000      4.000000      *
 1.000000      4.000000      *
 2.000000      6.000000      *
 3.000000      6.000000      *
 4.000000      6.000000      *
 5.000000      6.000000      *
```

EXAMPLE MLAB 3

```
*FUNCTION F(X)=X^2
*FUNCTION G(X)=SQRT(X)
*FUNCTION H(X)=2-X
*FUNCTION  K(X)=IF  X<0  THEN F(X) ELSE IF 0<=X AND X<1 THEN G(X)
```


CHAPTER IV.1 EXAMPLE MLAB 3
1
.5
0
-1
.5
2
Y=X12
Y=SQRT(X)
Y=2-X

```
ELSE H(X)
```

We will now graph the function K(X) which is constructed from pieces of F(X), G(X) and H(X).

```
*N=POINTS(K,-1:2:.1)
*DRAW N
*DISPLAY=T4012

*STRING A1="Y=X^2",AT -.8,.7,SIZE 4, ANGLE -72
*STRING A2="Y=SQRT(X)",AT .5,.75,SIZE 4,ANGLE 60
*STRING A3="Y=2-X",AT 1.6,.5,SIZE 4,ANGLE -70

*DELETE DEFAULTWINDOW
```

```
EXAMPLE MLAB 4

*FUNCTION S(X,Y)=IF X^2/9 + Y^2/4 < 1 THEN X*Y ELSE X+Y
*W COL 1=2:4:.2
*W COL 2=1:3:.2
*TYPE POINTS(S,W)

MATRIX :

2.000000 1.000000 2.000000 *
2.200000 1.200000 2.640000 *
2.400000 1.400000 3.800000 *
2.600000 1.600000 4.200000 *
2.800000 1.800000 4.600000 *
3.000000 2.000000 5.000000 *
3.200000 2.200000 5.400000 *
3.400000 2.400000 5.800000 *
3.600000 2.600000 6.200000 *
3.800000 2.800000 6.600000 *
4.000000 3.000000 7.000000 *
```

The function S(X,Y) defined above takes on the value X*Y if the point (X,Y) lies inside the ellipse centered at the origin with major axis equal to 3 and minor axis equal to 2 otherwise the function equals X+Y.

```
EXAMPLE MLAB 5
*FUNCTION G(X,Y)=IF X^2+Y^2<1 THEN 1 ELSE 2
*V COL 1=0:2:.2
*V COL 2=0:2:.2
*TYPE POINTS(G,V)

MATRIX :
```

```
.0000000 .0000000 1.000000 *
.2000000 .2000000 1.000000 *
.4000000 .4000000 1.000000 *
.6000000 .6000000 1.000000 *
.8000000 .8000000 2.000000 *
1.000000 1.000000 2.000000 *
1.200000 1.200000 2.000000 *
1.400000 1.400000 2.000000 *
1.600000 1.600000 2.000000 *
1.800000 1.800000 2.000000 *
2.000000 2.000000 2.000000 *
```

The function G(X,Y) defined above takes on the value 1 if the point (X,Y) lies inside the unit circle and takes on the value 2 otherwise.

IV.2 ROOTS OF EQUATIONS

The MLAB ROOT operator can be used to find the roots of general equations. The ROOT operator rather than simply producing a single number defines a function and the ROOT operator must be used in this functional definition form.

EXAMPLE 1.

```
FUNCTION F(X)=X^2-X-6
FUNCTION G(A,B)=ROOT(X,A,B,F(X))
TYPE G(0,5)
```

The function G defined in EXAMPLE 1 will look for roots of the function X^2-X-6 in the interval from A to B. If more than one root is found in the interval the root furthest to the left in the interval is given. The function F(X) has two roots at -2 and 3, so G(0,5)=3.

EXAMPLE 2.

```
FUNCTION K(T)=SIN(T)+COS(T)^2
FUNCTION H(A,B)=ROOT(T,A,B,K(T))
PI=3.14159
TYPE H(0,3*PI/2)
```

NOTE that the first argument of the ROOT operator agrees with the argument of the function K. We can interpret the root statement above as: find the value T in the interval from A to B for which K(T)=0; if no root is found in the interval MLAB will respond with a large negative number.

EXAMPLE 3.

```
FUNCTION L(X)=X^2-X*LOG(X)-5
FUNCTION M(A)=ROOT(X,1,A,L(X))
TYPE POINTS(M,2:5)
```

As with the INTEGRAL operator, we can use the ROOT operator to define a function of a single variable by fixing either the upper or the lower limit of the interval within which we are looking for roots. We chose as a lower limit 1 since L(1) is negative and the function eventually is positive and so must be 0 for some value of X greater than 1.

Perhaps the most practical way of determining the appropriate interval for the root operator is to graph the function and to read off rough values for the roots from the graph.

EXAMPLE MLAB 1

```
*FUNCTION F(X)=X^2+X-6
*FUNCTION G(A,B)=ROOT(X,A,B,F(X))
*TYPE G(-10,0),G(0,10),G(-10,10)

SCALAR = -3.000000

SCALAR = 2.000000

SCALAR = -3.000000
```

Note that the root selected on the full interval from -10 to 10 was the root furthest to the left, namely, -3.

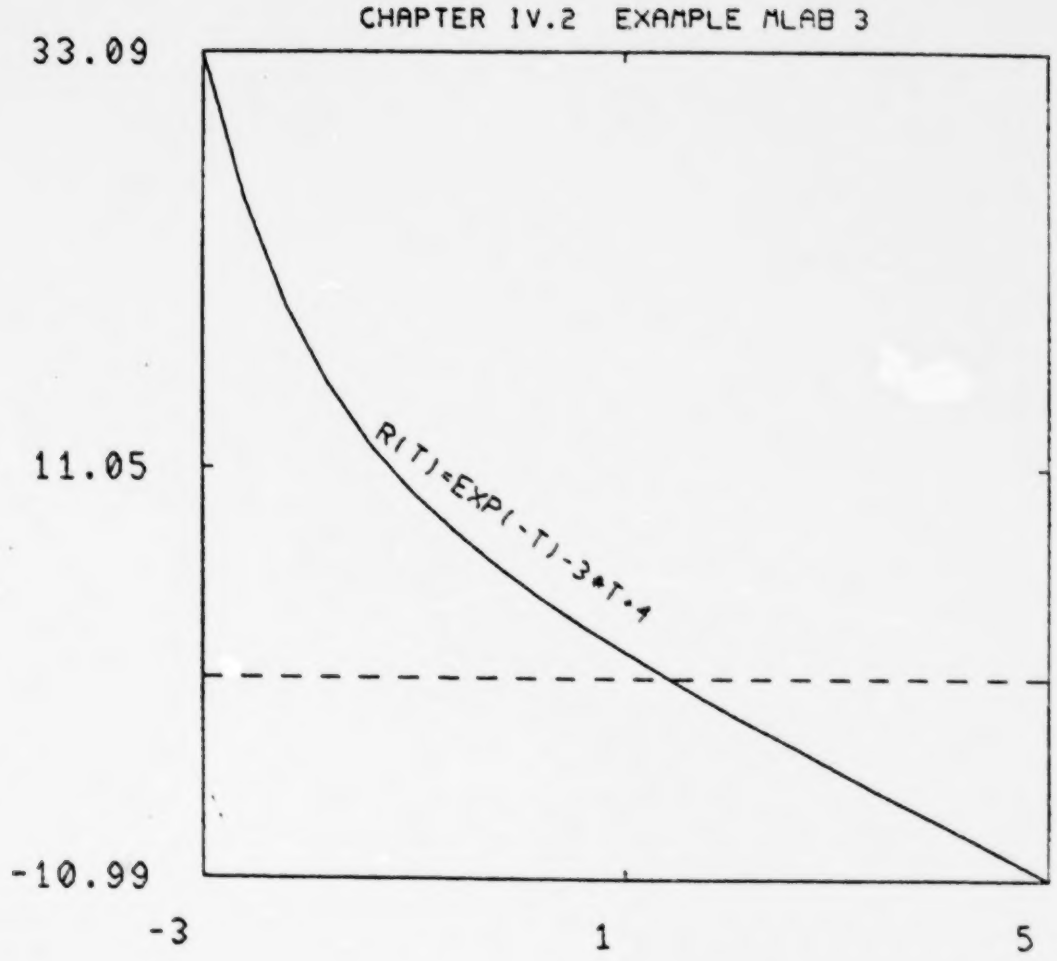
EXAMPLE MLAB 2

```
*FUNCTION T(X)=SIN(X)-COS(X)
*FUNCTION TROOT(A,B)=ROOT(X,A,B,T(X))
*TYPE TROOT(0,.5)
NOROOT

SCALAR = -.1701412@39
*TYPE TROOT(0,1)

SCALAR = .7853982
*TYPE T(.7853982)

SCALAR = .3725290@-7
```

In the above example we first asked for a root in the interval from 0 to .5; MLAB responded with NO ROOT and then a large negative number. We did find a root in the interval 0 to 1 and then as a check the original function was evaluated at the number which MLAB proposed as a root.

EXAMPLE MLAB 3

```
*FUNCTION R(T)=EXP(-T)-3*T+4

*N=POINTS(R,-3:5:.4)

*DRAW N

*DRAW (-3:5:.5),LINETYPE 2
```


CHAPTER IV.2 EXAMPLE MLAB 3
33.09
11.05
-10.99
-3
1
5
R(T)=EXP(-T)-3*T-4

```
*STRING A1="R(T)=EXP(-T)-3*T+4",AT -1.3,13,SIZE 3,ANGLE -35

*DISPLAY=T4012

*DELETE DEFAULTWINDOW
```

It is clear from the graph of R(T) that there is a root to the right of T=1. We drew the line Y=0 on the graph to help us approximate the root of R(T).

```
*FUNCTION S(B)=ROOT(T,0,B,R(T))
*TYPE POINTS(S,1:5)
NOROOT

MATRIX :

 1.000000    -.1701412@39  *
 2.000000     1.414360     *
 3.000000     1.414360     *
 4.000000     1.414360     *
 5.000000     1.414360     *
```

In this example we asked MLAB for the roots in the intervals 0 to 1, 0 to 2,..., 0 to 5. The only root found was at 1.414 and since the exponential part of R is decreasing more slowly than the linear part, 1.414 is the only root for T>0.

IV.3 THE FOR COMMAND AND THE SUM COMMAND

The FOR command permits the user of MLAB to construct loops. Its form is: FOR (a range of indices) DO (command(s) related to the indices).

EXAMPLE 1.

```
FOR K=1:5 DO(M(K)=SQRT(K))
TYPE M
```

In EXAMPLE 1 the range of indices is 1 to 5 and for each index K, M(K) is set equal to the square root of K. The TYPE command will cause the one column matrix M to be printed.

EXAMPLE 2.

```
PI=3.14159
FOR J=3:10 DO(S(J)=SIN(J*PI/4))
TYPE S
```

In EXAMPLE 2, a column vector S is defined. Since S(1) and S(2) are not explicitly defined MLAB will assign to each of these positions the value 0. Thus the TYPE S command will cause a one column vector with 10 entries to be typed out; the last 8 values will be SIN(3*PI/4), SIN(4*PI/4), ..., SIN(10*PI/4).

EXAMPLE 3.

```
PI=3.14159
FOR I=1:10 DO(FOR J=1:10 DO(C(I,J)=COS((I-J)*PI/20)))
FOR L=1:10 DO(TYPE C COL L)
```

In EXAMPLE 3, nested FOR commands are used to define a 10 by 10 matrix C. The last FOR command tells MLAB to type out the ten columns of the matrix C.

The MLAB SUM command is of the form SUM(I,A,B,G(I)) and simply tells MLAB to sum the values G(I) for indices I ranging from A to B. The SUM command is used to define a function of A and B.

EXAMPLE 4.

```
FUNCTION G(I)=I^2
FUNCTION F(A,B)=SUM(I,A,B,G(I))
TYPE F(11,20)
```

In EXAMPLE 4, the squares of the numbers from 11 to 20 are added together and typed out.

EXAMPLE 5.

```
PI=3.14159
FUNCTION H(J)=SIN(J*PI/8)
FUNCTION K(C,D)=SUM(J,C,D,H(J))
```

```
TYPE K(1,16)
```

In EXAMPLE 5,the index of summation is J and the sum of the sines of the multiples of PI/8 is computed.

EXAMPLE MLAB 1

```
*FOR I=1:5 DO(L(I)=I^3)
*M COL 1=1:5
*M COL 2=L
*TYPE M

MATRIX M:

1.000000 1.000000 *
2.000000 8.000000 *
3.000000 27.00000 *
4.000000 64.00000 *
5.000000 125.0000 *
```

In the above example, we use the FOR command to print out the values (X,X^3) for X=1,...,5.

EXAMPLE MLAB 2

```
*FOR I=1:4 DO(FOR J=1:4 DO(N(I,J)=I+J))
*TYPE N

MATRIX N:

2.000000 3.000000 4.000000 5.000000 *
3.000000 4.000000 5.000000 6.000000 *
4.000000 5.000000 6.000000 7.000000 *
5.000000 6.000000 7.000000 8.000000 *
```

The above example exhibits the use of nested FOR commands; up to 5 nested FOR loops can be used in a single command.

EXAMPLE MLAB 3

```
*FOR L=1:4 DO(TYPE N ROW L)

MATRIX :

2.000000 3.000000 4.000000 5.000000 *
```

```
MATRIX :

3.000000 4.000000 5.000000 6.000000 *

MATRIX :

4.000000 5.000000 6.000000 7.000000 *

MATRIX :

5.000000 6.000000 7.000000 8.000000 *
```

In this example we are using the matrix N which was defined in the previous example. The FOR command asks MLAB to type the rows of N in sequence.

```
EXAMPLE MLAB 4

*FUNCTION F(A,B)=SUM(I,A,B,I^2)
*TYPE F(1,5)

SCALAR = 55.00000
```

In this example F computes the sum 1^2 + 2^2 + ... + 5^2 and the TYPE F(1,5) causes the answer to be typed out.

```
EXAMPLE MLAB 5

*FUNCTION H(A,B)=SUM(J,A,B,J^3-J)
*TYPE H(1,100)

SCALAR = .2549745@8
```

In this example H(1,100) computes the sum (1^3-1) + (2^3-2) + (3^3-3) + ... + (99^3-99) + (100^3-100).

```
EXAMPLE MLAB 6

*X COL 1=LIST(7.1,8.3,5.5,6.9,5.4,7.1)
*FUNCTION AVE(N)=SUM(I,1,N,X(I))/N
*TYPE AVE(6)

SCALAR = 6.716667
*FUNCTION SS(N)=SUM(I,1,N,X(I)^2)/N
*TYPE SS(6)

SCALAR = 46.12167
```

In this example, AVE(N) computes the average of the first N numbers stored in the column vector X and AVE(6) is the average of all 6 values in X. Similarly SS(N) is the average sum of squares for the first N values stored in X.

CHAPTER V: FILES AND THE TEXT EDITOR

V.1 THE SAVE AND USE COMMANDS

The user of MLAB can store information from his session by use of the MLAB command SAVE. When the SAVE command is used a file is created and the designated information is placed in the file for use in future MLAB sessions.

The user can save his entire MLAB session by responding Y to the question at the beginning of the MLAB session which asks the user if he wishes to keep a log. In this case the entire MLAB session will be stored in a file named MLAB.LOG. The instruction at monitor level

```
.RENAME APEX=MLAB.LOG
```

will change the name of the MLAB.LOG file to APEX. This file is now treated in the same way as any file which is created using the TEXT EDITOR (see the next section).

EXAMPLE 1.

```
X=LIST(1.1,2.3,4.7,8.9,11.4)
SAVE X
```

The above SAVE command will create a file named X.DAT and the column vector X will be stored in X.DAT.

EXAMPLE 2.

```
USE X
TYPE X
```

In EXAMPLE 2, we are assuming that the user left MLAB after EXAMPLE 1 and has signed on again. The USE X command will bring the file X.DAT and its contents X into MLAB. The TYPE X command will cause X to printed out.

EXAMPLE 3.

```
W COL 1=LIST(1,2,3,4,5)
W COL 2=LIST(-1.1,2.3,1.9,-3.4,-5.6)
FUNCTION F(X,Y)=1-X^2+2*X*Y
SAVE W IN PLAN1
SAVE F IN PLAN1
```

In EXAMPLE 3 we are storing more than one entry and rather than store each quantity in its own file we choose to save both W and F in the same file, which we name PLAN1. MLAB then adds the extension DAT, so the file's name is PLAN1.DAT.

EXAMPLE 4.

```
USE PLAN1
TYPE W
TYPE F
```

In EXAMPLE 4, we assume that the MLAB user has exited from MLAB after EXAMPLE 3 and now has reentered MLAB. After the command USE PLAN1, MLAB will type out

```
USING W,F
```

and thus informs the user of the contents of the file named PLAN1.DAT. The TYPE W and TYPE F commands are given to see the contents of W and F.

```
EXAMPLE MLAB 1

*N COL 1=LIST(1.2,2.2,3.2,4.2,5.1)
*N COL 2=LIST(-1,-2.3,-2.7,-3.4,-3.8)
*N COL 3=LIST(7.3,6.5,4.4,3.9,2.5)
*SAVE N
*FUNCTION F(X,Y)=A+B*X+C*Y
*A=1;B=1;C=1
*CONSTRAINTS T
!A>0
!A+2*B>0
!
*QUIET FIT(A,B,C),F TO N;!T;!4;!;

FINAL                NORMAL ERROR         DEPENDENCY
PARAMETER VALUES:    STANDARD ERRORS:     VALUES:
     8.58408             .627965          .909785     A
    -1.68130             .630703          .992568     B
    -.637288             .898708          .994436     C
LAGRANGE MULTIPLIERS:
          .000000
          .000000

CONVERGED
RMS ERROR= .421756
FINAL SUM OF SQUARES= .355756
# OF ITERATIONS USED=3

*SAVE A,B,C,T,F,N IN FITPL
```

In this example, we enter a 3 column data matrix named N and ask MLAB to find the best planar fit F(X,Y) to the given data matrix. We create a file named N.DAT to hold the matrix N and we create a file named FITPL.DAT which contains the final values of the parameters A,B,C, the constraints T, the function F and the matrix N.

We used the QUIET FIT to suppress the intermediate output of the fitting routine. Appended to the QUIET FIT statement are the responses to the questions concerning constraints, number of iterations and the convergence factor. The sequence !T;!4;!; represents CONSTRAINTS T,4 iterations and the default value for the convergence factor.

EXAMPLE MLAB 2

```
*USE N
*TYPE N

MATRIX N:

1.200000 -1.000000 7.300000 *
2.200000 -2.300000 6.500000 *
3.200000 -2.700000 4.400000 *
4.200000 -3.400000 3.900000 *
5.100000 -3.800000 2.500000 *

*USE FITPL
USING:A,B,C,T,F,(REPLACING)N
*TYPE A,B,C,T,F

SCALAR A = 8.584083
SCALAR B = -1.681296
SCALAR C = -.6372878

CONSTRAINTS T
A>0
A+2*B>0

FUNCTION F(X,Y) = A+B*X+C*Y
```

We signed off from MLAB after doing the EXAMPLE MLAB 1 above. After reentering MLAB, we did this example and then employed the USE command to bring the files N.DAT and FITPL.DAT into MLAB. These two files are still being stored on a disk; to destroy the files we must go outside of MLAB.

V.2 THE TEXT EDITOR

The text editor used in conjunction with MLAB is named SOS and there is available a manual containing all the SOS instructions and commands. In this brief discussion of the text editor, we introduce a few basic commands:

```
P: THE PRINT COMMAND
D: THE DELETE COMMAND
I: THE INSERT COMMAND
S: THE SUBSTITUTE COMMAND
E: THE END OF FILE COMMAND
```

These commands are the minimum needed for constructing and editing files. In a file we speak of an input mode and an edit mode. The edit mode is signaled with an * as a prompt and the input mode is signaled by a line number.

When the user of MLAB logs on a . is received as a prompt. Then to enter MLAB the following is typed

```
.RUN PRO:MLAB
```

and the user proceeds to use MLAB. The prompt . indicates that the user is at monitor level. Instead of using the RUN PRO:MLAB command, the user can type

```
.SOS REGRS
```

This command will access a file named REGRS if it exists; if no such file exists, a file named REGRS is created.

EXAMPLE 1.

```
.SOS REGRS
EDIT: REGRS
*
```

The above command calls for a file named REGRS; the response EDIT:REGRS indicates that such a file exists. The user is then prompted with * and an editing instruction is awaited.

EXAMPLE 2.

```
.SOS MINV
INPUT: MINV
100
```

The SOS command in this case called for a file named MINV. The response INPUT rather than EDIT indicates that a file named MINV did not exist before. The user is then prompted with a line numbered 100 and input is expected.

EXAMPLE 3.

```
.SOS MINV
INPUT: MINV
100 THIS FILE CONTAINS A DESCRIPTION OF  VARIOUS  METHODS
200 USED TO INVERT MATRICES.
300 .........
400 .........
.
.
.
5500 AND MANY OTHERS.
5600$
*E
.
```

In the above example, the line 100 was given to us as a prompt and we entered the text indicated. At the end of line 100 we pressed a carriage return and received the line 200 as a prompt. The lines are automatically incremented by 100 as long as we are entering data and information into the file.

We finished our text on line 5500 and wished to stop entering quantities in the file . At the end of 5500 we pressed a carriage return. After the prompt 5600 we pressed the escape key and received the prompt *, to which we responded E to end the file. Finally the prompt . indicates that we are again at monitor level.

EXAMPLE 4.

```
.SOS REGRS
EDIT: REGRS
*P0:*
```

In response to the prompt * we typed the command P0:*, which simply requests that the file be printed out. The edit command PA:B means print out all lines in the file from line A through line B. Each line in a file has a line number. In the command P0:*, the * stands for the number of the last line in the file.

EXAMPLE 5.

```
.SOS REGRS
EDIT: REGRS
*P100:2000
.
.
.
*I1050,10
1050
```

In this example, we respond P100:2000 to the initial * prompt. Those lines are automatically printed out and we again receive * as a prompt. This time we respond I1050,10 which means insert the

line 1050 and continue inserting lines in increments of 10. Then we receive a prompt of 1050 and this line is awaiting input. We will continue receiving line input prompts until the line 1100 is reached; at this point we will be placed in edit mode and will receive an * as a prompt.

The commands

```
*D150
*D3570:4300
```

in edit mode simply request that line 150 be deleted and that all the lines between 3570 and 4300 be deleted. The command

```
*SOLD TEXT$NEW TEXT$7200
```

in edit mode substitutes for old text, new text in line 7200. The $ in this case corresponds to pressing the escape key.

Pressing the escape key will take the user from input mode to edit mode. Using the insert command I will take the user from edit mode to input mode.

V.3 DO FILES

DO FILES are files consisting of MLAB executable statements. They are constructed using the text editor and cannot be constructed using MLAB.

Statements in DO FILES are exactly as they appear in MLAB except that each line must end with a semicolon.

EXAMPLE MLAB 1

```
.SOS POLFIT
INPUT: POLFIT
100 TYPE "THIS FILE EXPECTS AN INPUT DATA MATRIX NAMED
M";
200 TYPE "THEN POLFIT WILL FIT A CUBIC POLNOMIAL TO M";
300 TYPE "NOW ENTER M.";
400 DO("TTY:");
500 DO("TTY:");
600 FUNCTION F(X)=A+B*X+C*X^2+D*X^3;
700 A=1;B=1;C=1;D=1;
800 QUIET FIT(A,B,C,D),F TO M;!;!10;!;
900 TYPE F,A,B,C,D;
1000$
*E
```

The file POLFIT above can be called into MLAB via the command

```
DO("POLFIT")
```

After the above command the verbal comments enclosed in quotes in lines 100 to 300 will be printed out. The DO("TTY:") command will cause the MLAB input prompt * to appear. At this point the MLAB user is expected to enter M COL 1. A carriage return then puts the user back into the DO FILE at statement 500. The user will get another MLAB input prompt and should respond by entering M COL 2.

Now that M is defined the DO FILE will cause the indicated fit to be performed and will type out F,A,B,C and D. Then the instructions in the file having been completed, the user will receive an MLAB input prompt and all conversation with the DO FILE POLFIT is terminated.

EXAMPLE MLAB 2

```
*DO("POLFIT")
THIS FILE EXPECTS AN INPUT DATA MATRIX NAMED M
THEN POLFIT WILL FIT A CUBIC POLYNOMIAL TO M
NOW ENTER M.
*M COL 1=1:5:.5
*M COL 2=LIST(-1,.2,.6,.8,.2,-.4,-.7,-.3,.2)
```

```
FINAL                NORMAL ERROR        DEPENDENCY
PARAMETER VALUES:    STANDARD ERRORS:    VALUES:
    -6.89291             .650783         .993936     A
     8.59227             .803833         .999627     B
    -3.04340             .292265         .999829     C
      .322251            .322448@-1      .999281     D

CONVERGED
RMS ERROR= .152028
FINAL SUM OF SQUARES= .115563
# OF ITERATIONS USED=8

FUNCTION F(X) = A+B*X+C*X^2+D*X^3
SCALAR A = -6.892910
SCALAR B = 8.592274
SCALAR C = -3.043398
SCALAR D = .3222513
*EXIT
```

EXAMPLE MLAB 3

```
.SOS CAUCHY
 INPUT: CAUCHY
 100 TYPE "THIS DO FILE FITS A FUNCTION OF THE FORM";
 200 TYPE "         F(X)=B/(X^2+A^2)          ";
 300 TYPE " TO AN INPUT MATRIX NAMED D. THE USER MUST INPUT";
 400 TYPE " THE TWO COLUMN MATRIX D AND MUST INPUT THE INITIAL";
 500 TYPE " VALUES FOR A AND B.";
 600 TYPE "NOW ENTER THE VALUES FOR D COL 1.";
 700 DO("TTY:");
 800 TYPE " NOW ENTER THE VALUES FOR D COL 2.";
 900 DO("TTY:");
1000 FUNCTION F(X)=B/(X^2+A^2);
1100 TYPE " NOW INITIALIZE A AND B.";
1200 DO("TTY:");
1300 QUIET FIT(A,B),F TO D;!;!10;!;
1400 M=D&'(F ON D);
1500 TYPE M;
1600 DRAW M COL(1,2),POINTTYPE 3,LINETYPE 2;
1700 DRAW M COL(1,3);
1800 DISPLAY=T4012;
```

The above DO FILE named CAUCHY can be called into MLAB with the command DO("CAUCHY") or DO "CAUCHY". This DO FILE fits a curve of the form

$$F(X)=B/(X^2+A^2)$$

to input data. When CAUCHY is called into MLAB the expressions enclosed in quotes will be printed out as an aid to the user. As in the preceding example the DO("TTY:") is used to transfer to

the terminal so that the user may enter data or issue MLAB commands. The next example illustrates the use of this DO FILE.

```
EXAMPLE MLAB 4

*DO "CAUCHY"

 THIS DO FILE FITS A FUNCTION OF THE FORM

         F(X)=B/(X^2+A^2)

  TO AN INPUT MATRIX NAMED D. THE USER MUST INPUT

  THE TWO COLUMN MATRIX D AND MUST INPUT THE INITIAL

  VALUES FOR A AND B.

 NOW ENTER THE VALUES FOR D COL 1.

*D COL 1=-2:2:.4

  NOW ENTER THE VALUES FOR D COL 2.

*D COL 2=LIST(.1,.2:1:.2,.8:.2:-.2,.1)

  NOW INITIALIZE A AND B.

*A=1;B=1


FINAL                NORMAL ERROR        DEPENDENCY
PARAMETER VALUES:    STANDARD ERRORS:    VALUES:
     .903190           .519047@-1        .917185     A
     .808280           .743001@-1        .917185     B

CONVERGED
RMS ERROR= .488665@-1
FINAL SUM OF SQUARES= .214914@-1
# OF ITERATIONS USED=3
# ACTUALS NEQ # FORMALS FOR F

MATRIX :

-2.000000      .1000000      .1678408      *
-1.600000      .2000000      .2394369      *
-1.200000      .4000000      .3583193      *
-.8000000      .6000000      .5552315      *
-.4000000      .8000000      .8283653      *
 .0000000     1.000000       .9908392      *
 .4000000      .8000000      .8283654      *
 .8000000      .6000000      .5552315      *
 1.200000      .4000000      .3583193      *
```

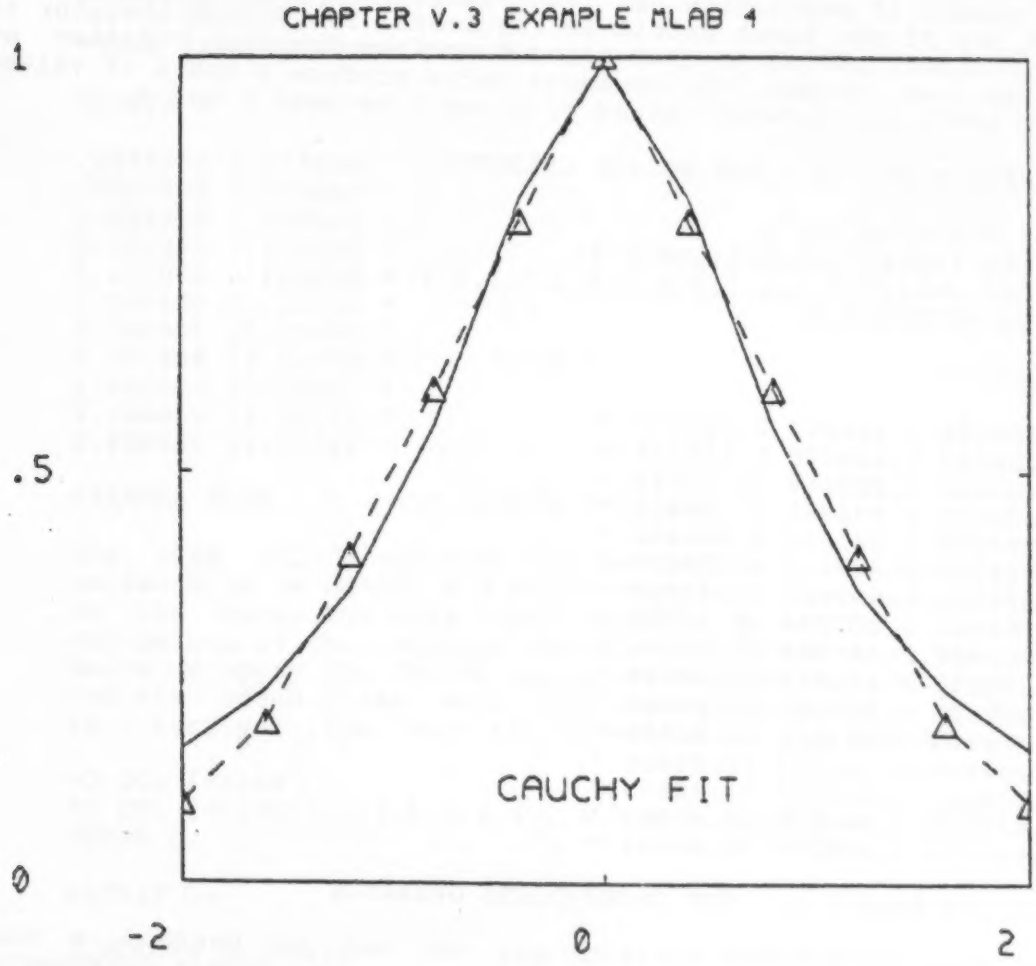
```
 1.600000      .2000000      .2394369      *
 2.000000      .1000000      .1678408      *
```

At this point the DRAW command in the DO FILE will execute and the graph of the input data will appear as a dashed line and the graph of the fitting function as a solid curve. We add the following commands in the MLAB session to adjust the scaling of the vertical axis and to label the graph.

```
*DRAW (-2:2:.5)&'0
*STRING A1="CAUCHY FIT",AT -.5,.12,SIZE 4
*DELETE DEFAULTWINDOW
```

See the accompanying graph for the final result of these commands.

```
*EXIT
```


CHAPTER V.3 EXAMPLE MLAB 1
1
.5
0
-2
0
2
CAUCHY FIT

CHAPTER VI: SOME DETAILED EXAMPLES

All the examples in this chapter contain MLAB input, MLAB output and explanatory comments.

EXAMPLE MLAB 1. - THE POINTS, INTERPOLATE AND SMOOTH OPERATORS

The following MLAB dialog illustrates the use of the MLAB POINTS INTERPOLATE, and SMOOTH operators. The MLAB POINTS operator acts on a function and a matrix, and returns a matrix with one more column than the input matrix. Each row of the output matrix contains The elements of that row of the input matrix followed by the result of evaluating the input function using the elements of that row of the input matrix as arguments. The POINTS operator is particularly useful in evaluating functions for the creation of tables and graphs. the commands below produce a table of values of X^2+Y^2 for integer values of X and Y between 1 and 4.

EXAMPLE MLAB 1.A - THE POINTS OPERATOR

```
*FCT F(X,Y)=X^2+Y^2
*M COL 1=LIST(1:4,1:4,1:4,1:4)
*M COL 2=LIST(1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4)
*TYPE POINTS(F,M)

MATRIX :

1.000000 1.000000 2.000000 *
2.000000 1.000000 5.000000 *
3.000000 1.000000 10.00000 *
4.000000 1.000000 17.00000 *
1.000000 2.000000 5.000000 *
2.000000 2.000000 8.000000 *
3.000000 2.000000 13.00000 *
4.000000 2.000000 20.00000 *
1.000000 3.000000 10.00000 *
2.000000 3.000000 13.00000 *
3.000000 3.000000 18.00000 *
4.000000 3.000000 25.00000 *
1.000000 4.000000 17.00000 *
2.000000 4.000000 20.00000 *
3.000000 4.000000 25.00000 *
4.000000 4.000000 32.00000 *
```

EXAMPLE MLAB 1.B - THE INTERPOLATE OPERATOR

The MLAB INTERPOLATE operator may be use to perform a cubic spline interpolation between data points, to obtain intermediate values of a function given only by point data. We input a matrix N containing data from 0 to 5 at intervals of 1 and use the interpolate operator to obtain values at intervals of .5. The arguments of the INTERPOLATE operator are a two column matrix of data and a one column matrix of values at which the interpolated value of the function represented by the data is desired.

```
*N COL 1=0:5
*N COL 2= LIST(1,3,6,10,15,10)
*TYPE N

MATRIX N:

.0000000 1.000000 *
1.000000 3.000000 *
2.000000 6.000000 *
3.000000 10.00000 *
4.000000 15.00000 *
5.000000 10.00000 *

*TYPE INTERPOLATE(N,0:5:.5)

MATRIX :

.0000000 1.000000 *
.5000000 1.875000 *
1.000000 3.000000 *
1.500000 4.375000 *
2.000000 6.000000 *
2.500000 7.926136 *
3.000000 10.00000 *
3.500000 12.50000 *
4.000000 15.00000 *
4.500000 14.26136 *
5.000000 10.00000 *
```

EXAMPLE MLAB 1.C - THE SMOOTH OPERATOR

The MLAB SMOOTH operator may be used to SMOOTH out data that is believed to be noisy. A 5 point hyperbolic smoothing is performed on the data. The only input argument is a matrix of values. The end-points of the returned matrix are generally untrustworthy. Below we apply the SMOOTH operator to data representing the curve Y=X with added noise. Note that the returned curve is very close to a straight line, with the exception of the end-points.

```
*O COL 1=1:10
*O COL 2=LIST(1.2,1.9,3.0,4.1,4.8,6.2,7.1,8.0,9.1,10.2)
*TYPE O

MATRIX O:

1.000000 1.200000 *
2.000000 1.900000 *
3.000000 3.000000 *
4.000000 4.100000 *
5.000000 4.800000 *
6.000000 6.200000 *
7.000000 7.100000 *
8.000000 8.000000 *
9.000000 9.100000 *
10.00000 10.20000 *
```

```
*P=SMOOTH(O)
*TYPE P

MATRIX P:

1.000000 1.435252 *
2.000000 2.097940 *
3.000000 3.000000 *
4.000000 4.014134 *
5.000000 4.989541 *
6.000000 6.071519 *
7.000000 7.063746 *
8.000000 8.085230 *
9.000000 9.063522 *
10.00000 9.881871 *

*DRAW O

*DRAW P
*DISPLAY=T4010
```

EXAMPLE MLAB 2. - THE FIT COMMAND

The following MLAB dialog illustrates the use of MLAB to fit data that is a function of 2 independent variables to a model. In this case we enter data for all combinations of integer values of X and Y between 0 and 3. We fit the data to a curve of the form F(X,Y)=AX+BX^2+CY+DY^2.

```
*M COL 1=LIST(0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3)
*M COL 2=LIST(0:3,0:3,0:3,0:3)
*M COL 3=LIST(0,2.8,5.8,10.6,2,2.7,8,12.3,6,8.6,12.3,18,11,14,18,
22)
*A=0
*B=1
*C=0
*D=1
*FCT F(X,Y)=A*X+B*X^2+C*Y+D*Y^2
*QUIET FIT(A,B,C,D),F TO M;!;!6;!

FINAL                NORMAL ERROR        DEPENDENCY
PARAMETER VALUES:    STANDARD ERRORS:    VALUES:
     1.05637              .475007         .959913     A
     .940151              .163893         .951895     B
     1.67884              .475007         .959912     C
     .652659              .163893         .951895     D

CONVERGED
RMS ERROR= .711702
FINAL SUM OF SQUARES= 6.07824
# OF ITERATIONS USED=2
```

We use the points operator to set up a matrix N which contains the X and Y values in columns 1 and 2, the values given by the model in column 3 and the data in column 4.

```
*N=POINTS(F,M COL(1,2))
*N COL 4=M COL 3
*TYPE N

MATRIX N:

.0000000 .0000000 .0000000 .0000000 *
.0000000 1.000000 2.331501 2.800000 *
.0000000 2.000000 5.968321 5.800000 *
.0000000 3.000000 10.91046 10.60000 *
1.000000 .0000000 1.996518 2.000000 *
1.000000 1.000000 4.328019 2.700000 *
1.000000 2.000000 7.964838 8.000000 *
1.000000 3.000000 12.90698 12.30000 *
2.000000 .0000000 5.873338 6.000000 *
2.000000 1.000000 8.204839 8.600000 *
2.000000 2.000000 11.84166 12.30000 *
2.000000 3.000000 16.78380 18.00000 *
3.000000 .0000000 11.63046 11.00000 *
3.000000 1.000000 13.96196 14.00000 *
3.000000 2.000000 17.59878 18.00000 *
3.000000 3.000000 22.54092 22.00000 *
```

EXAMPLE MLAB 3. - DIFFERENTIAL EQUATIONS AND THE FIT COMMAND

The following example illustrates the use of the MLAB curve-fitting routines to determine both an unknown parameter and an unknown initial value in a differential equation model, by fitting the model to data. Let us suppose that we have some data that we believe can be fit using a simple exponential population growth model. The differerential equation for this model is

dY/dT=AY with initial value Y(0)=B.

We shall use MLAB to obtain values for A and B which best fit the data in a least-squares sense.

First we enter the data into a matrix M, using the LIST operator.

```
*M COL 1=0:4:.2
*M COL 2=LIST(1.3,1.3,1.4,1.5,1.5,1.7,1.8,2.0,2.4,2.8,3.2,3.6,4.1
,4.6,5.2,5.8,6.3,7,8.2,8.8,9.9)
*TYPE M

MATRIX M:

.0000000 1.300000 *
.2000000 1.300000 *
.4000000 1.400000 *
.6000000 1.500000 *
```

```
.8000000 1.500000 *
1.000000 1.700000 *
1.200000 1.800000 *
1.400000 2.000000 *
1.600000 2.400000 *
1.800000 2.800000 *
2.000000 3.200000 *
2.200000 3.600000 *
2.400000 4.100000 *
2.600000 4.600000 *
2.800000 5.200000 *
3.000000 5.800000 *
3.200000 6.300000 *
3.400000 7.000000 *
3.600000 8.200000 *
3.800000 8.800000 *
4.000000 9.900000 *
```

Now we enter the differential equation model and guess values for the parameters A and B. We take as our guess values A=1 and B=1. Some care must be taken in the selection of guess values, as the curve fitting routines may locate a local minimum that is not the true minimum, if guess values that are too far from the best fit are chosen.

```
*FCT Y DIFF T(T)=A*Y
*INITIAL Y(0)=B
*A=1
*B=1
*QUIET FIT(A,B),Y TO M
```

The FIT command tells MLAB to perform the fitting operation. The quiet adjective suppresses detailed output from the fit. MLAB now asks for derivative names, constraints, maximum number of iterations, and convergence factor. All of these could have been entered with the FIT command itself, if the user so desired.

```
TYPE DERIVATIVE NAME(S) OF DEQ SYSTEM FOR MODEL
Y DIFF T

CONSTRAINTS NAME=(NONE)?
MAX NUMBER OF ITERATIONS=(3)? 6
CONVERGENCE FACTOR FOR SUM OF SQUARES =(.001)?

FINAL                NORMAL ERROR        DEPENDENCY
PARAMETER VALUES:    STANDARD ERRORS:    VALUES:
     .569110           .884319@-2        .944803     A
     1.02575           .303661@-1        .944803     B

CONVERGED
RMS ERROR= .153389
FINAL SUM OF SQUARES= .447034
# OF ITERATIONS USED=6
```

MLAB returns the values .57 for A and 1.02 for B. Now we use the MLAB INTEGRATE operator to integrate the differential equation using these values so that we may compare the model with the data.

```
*O=INTEGRATE(Y DIFF T,0:4:.2)
*O COL 3=M COL 2
```

We set column 3 of the matrix O equal to column 2 of M. O then contains both data and model.

```
*TYPE O

MATRIX O:

.0000000 1.025750 1.300000 *
.2000000 1.149410 1.300000 *
.4000000 1.287976 1.400000 *
.6000000 1.443244 1.500000 *
.8000000 1.617229 1.500000 *
1.000000 1.812191 1.700000 *
1.200000 2.030654 1.800000 *
1.400000 2.275454 2.000000 *
1.600000 2.549766 2.400000 *
1.800000 2.857144 2.800000 *
2.000000 3.201582 3.200000 *
2.200000 3.587539 3.600000 *
2.400000 4.020023 4.100000 *
2.600000 4.504649 4.600000 *
2.800000 5.047693 5.200000 *
3.000000 5.656204 5.800000 *
3.200000 6.338074 6.300000 *
3.400000 7.102139 7.000000 *
3.600000 7.958325 8.200000 *
3.800000 8.917718 8.800000 *
4.000000 9.992760 9.900000 *
```

We use the MLAB DRAW command to obtain graphs of both the solution and the data.

```
*DRAW O COL (1,2)

*DRAW O COL (1,3)
*DISPLAY=T4010
```

In order to get a better picture of the accuracy of our model, we create a fourth column of O, containing the differences between data and model. We type out and draw this curve of errors.

```
*O COL 4=O COL 2-O COL 3
*TYPE O COL(1,4)

MATRIX :
```

```
 .0000000    -.2742505     *
 .2000000    -.1505896     *
 .4000000    -.1120245     *
 .6000000    -.5675644@-1  *
 .8000000     .1172293     *
1.000000      .1121912     *
1.200000      .2306538     *
1.400000      .2754542     *
1.600000      .1497660     *
1.800000      .5714417@-1  *
2.000000      .1582176@-2  *
2.200000    -.1246104@-1   *
2.400000    -.7997739@-1   *
2.600000    -.9535086@-1   *
2.800000    -.1523073      *
3.000000    -.1437962      *
3.200000     .3807396@-1   *
3.400000     .1021388      *
3.600000    -.2416751      *
3.800000     .1177176      *
4.000000     .9275973@-1   *
```

```
*DELETE DEFAULTWINDOW
*DRAW O COL (1,4)
```

We see that the errors are small and do not appear to be systematic. If we were dealing with actual data, this would indicate that our model was reasonable.

EXAMPLE MLAB 4. -- SOLVING SYSTEMS OF NON-LINEAR EQUATIONS

The following example illustrates the use of the MLAB curve-fitting routines to solve a set of non-linear simultaneous equations. Consider the following system of equations:

```
X^2+Y^2+Z^2=1

Z=X^2+Y^2

Y=X^3
```

The first equation is the equation of a sphere of radius 1 about the origin. The second equation is the equation of a paraboloid with axis along the Z axis and minimum Z value at the origin. The third equation is a surface parallel to the 2 dimensional curve Y=X^3. Consideration of the geometry of the problem leads to the conclusion that there will be 2 roots, one with X, Y, and Z positive, and one with X and Y negative and Z positive.

In order to use the MLAB curve-fitting routines to solve these equations, we first move all the terms in each equation to the left-hand-sides, and set up 3 functions, F, G, and H, of a dummy argument D, which are equal to the left-hand-sides of the 3 equations. We will fit all three functions simultaneously to the single data point (0,0). The effect of this will be to minimize the sum of the squares of the left-hand-sides of the equations. As all three left-hand-sides should be zero at a solution of the equation, the fit should converge to a solution for X, Y, and Z and give a value for the final sum of the squares that is essentially zero. If there were no root, the fit should converge to values for X, Y, and Z which minimize the sum of the squares of the l.h.s.'s. As this is a non-linear fit, the results can be guess dependent, and a poorly chosen guess may cause the fit to converge to a local minimum that is quite far from an actual root.

```
*FCT F(D)=X^2+Y^2+Z^2-1
*FCT G(D)=Z-X^2-Y^2
*FCT H(D)=Y-X^3
*M=LIST(0,0)
```

We shall first try to find the root with X, Y, and Z positive. We take as our guess X=1, Y=1, Z=1.

```
*X=1
*Y=1
*Z=1

*QUIET FIT(X,Y,Z),F TO M,G TO M,H TO M;!;!12;!

FINAL                NORMAL ERROR         DEPENDENCY
PARAMETER VALUES:    STANDARD ERRORS:     VALUES:
     .704277           .273456@-8        .243729@-1 X
     .349325           .483265@-8        .227714@-1 Y
     .618034           .424748@-8        .110228@-1 Z

CONVERGED
RMS ERROR= .671586@-8
FINAL SUM OF SQUARES= .451028@-16
# OF ITERATIONS USED=6
```

We see that the final sum of the squares is almost zero and thus conclude that we have found a root. Now we take the guess X=-1 Y=-1, Z=.618 and attempt to locate the root for which X and Y are negative.

```
*X=-1
*Y=-1
*QUIET FIT(X,Y,Z),F TO M,G TO M,H TO M;!;!12;!
```

```
FINAL                NORMAL ERROR        DEPENDENCY
PARAMETER VALUES:    STANDARD ERRORS:    VALUES:
    -.704277           .273456@-8        .243729@-1 X
    -.349325           .483265@-8        .227714@-1 Y
     .618034           .424748@-8        .110228@-1 Z

CONVERGED
RMS ERROR= .671586@-8
FINAL SUM OF SQUARES= .451028@-16
# OF ITERATIONS USED=7
```

We see that the final sum of the squares is very small and that we have found the other root. In a more complicated example there might be many roots, or no root at all. The user might not be sure how many roots there were or where the meaningful roots might be located. Some careful playing around with the guess may be required to locate all roots of interest. For any particular example, the result of the curve fit, be it a root or a local minimum, will be a function of the guess used.

EXAMPLE MLAB 5. - A SECOND ORDER DIFFERENTIAL EQUATION

The following example illustrates the MLAB procedure for solving second or higher order ordinary differential equations. It also illustrates the solution of boundary-value problems in ordinary differential equations by using the MLAB fit statement. Consider the differential equation d^2Y/dT^2=-Y, with boundary conditions Y(Ø)=Ø and Y(1)=1. MLAB is capable of integrating only first order differential equation systems, so the first step is to rewrite the second order equation as two first order equations by defining Y1=dY/dT, and substituting in the original equation to get the equation dY1/dT=-Y.

```
*FCT Y DIFF T(T)=Y1
*FCT Y1 DIFF T(T)=-Y
```

The MLAB differential equation solver requires initial values for each of the dependent variables in the system. We set Y(Ø) equal to Ø and the initial value of Y1(Ø) to the unknown parameter A. We now fit the system of equations to the single data point (1,1). As we are fitting 1 data point to 1 unkown, we expect an exact solution. We make the initial guess A=1.

```
*INITIAL Y(Ø)=Ø
*INITIAL Y1(Ø)=A
*A=1
*M = LIST(1,1)
*QUIET FIT(A),Y TO M
```

MLAB recognizes that the function Y is defined by a differential equation, and asks the user to enter all derivatives used in the deq system of the model. This information could have been entered

in one line along with the fit statement.

```
TYPE DERIVATIVE NAME(S) OF DEQ SYSTEM FOR MODEL
Y DIFF T,Y1 DIFF T

CONSTRAINTS NAME=(NONE)?
MAX NUMBER OF ITERATIONS=(3)? 10
CONVERGENCE FACTOR FOR SUM OF SQUARES =(.001)?

FINAL                NORMAL ERROR        DEPENDENCY
PARAMETER VALUES:    STANDARD ERRORS:    VALUES:
     1.18839           .000000           .000000    A

CONVERGED
RMS ERROR= .000000
FINAL SUM OF SQUARES= .000000
# OF ITERATIONS USED=3
```

MLAB returns the value 1.19 for A. The final sum of squares is 0, as we would expect when using the curve fitter to solve an exact problem rather than using it to fit a model to experimental data. We use the INTEGRATE statement to observe the behavior of our differential equation for T values from 0 to 4, using the initial value for Y1 obtained from the fit.

```
*N=INTEGRATE(Y DIFF T,Y1 DIFF T,0:4:.2)
```

The INTEGRATE statement returns a matrix whose first column contains values of the independent variable; the next two columns contain the values of the first dependent variable and its derivative; the final two columns contain values of the second dependent variable and its derivative etc. We can see the equations in the deq system illustrated by the values of the table. We next draw curves for both Y and Y1. They appear to be out-of-phase sinusoids of period 2*PI, as we would expect from the mathematics of the system under consideration.

```
*TYPE N

MATRIX N:

 .0000000     .0000000     1.188394     1.188394     .0000000     *
 .2000000     .2361040     1.164708     1.164708    -.2361040     *
 .4000000     .4627860     1.094591     1.094591    -.4627860     *
 .6000000     .6710138     .9808269     .9808269    -.6710138     *
 .8000000     .8524981     .8279751     .8279751    -.8524981     *
1.000000      .9999933     .6421088     .6421088    -.9999933     *
1.200000     1.107617      .4306430     .4306430   -1.107617      *
1.400000     1.171102      .2020177     .2020177   -1.171102      *
1.600000     1.187888    -.3466578@-1 -.3466578@-1 -1.187888      *
1.800000     1.157318     -.2699648    -.2699648   -1.157318      *
2.000000     1.080624     -.4945040    -.4945040   -1.080624      *
```

```
 2.200000     .9608385     -.6993257     -.6993257     -.9608385      *
 2.400000     .8027558     -.8762705     -.8762705     -.8027558      *
 2.600000     .6126727     -1.018286     -1.018286     -.6126727      *
 2.800000     .3981601     -1.119698     -1.119698     -.3981601      *
 3.000000     .1677811     -1.176484     -1.176484     -.1677811      *
 3.200000    -.6928900@-1 -1.186365     -1.186365      .6928900@-1   *
 3.400000    -.3035950     -1.148944     -1.148944      .3035950      *
 3.600000    -.5258003     -1.065737     -1.065737      .5258003      *
 3.800000    -.7270422     -.9400330     -.9400330      .7270422      *
 4.000000    -.8992952     -.7768519     -.7768519      .8992952      *

*DELETE DEFAULTWINDOW
*DRAW N COL(1,2)

*DRAW N COL(1,4)
*GET DISPLAY
```

EXAMPLE MLAB 6. - THE READ AND PRINT STATEMENTS

EXAMPLE MLAB 6.A - THE READ STATEMENT

The MLAB READ operator may be used to input numerical data stored on disk data sets. This data may be created by any method desired. possiblities include entering data into a file manually using the text editor, writing the data to the file using MLAB, writing the data to the file using a FORTRAN program written on the ADP system, or writing a tape using another system and sending it to ADP for entry into the system. The data can consist of a series of numbers, which may be separated by blanks, commas, or virtually any other non-numeric character. The numbers may be in integer or decimal form, or in decimal form with an exponent. If they are in the latter form, the character @ must be used for the exponent instead of the character E as in FORTRAN. files produced by FORTRAN E-format write statements may be converted easily using the text editor.

The following is a listing of a file, called PRD1, which will be read into MLAB in various forms.

```
6 6.777 2.2 0.0000054 .543@11
1 3,6,6 9 999 3
```

Note that the numbers are in various formats, and spaced by blanks or commas. The MLAB READ expression is in the form READ("filename",N,C) where filename is the name of the file being read, N is the number of values to be read out of the file, and C is the number of columns in the matrix returned by the read operator. If C is omitted, a value of 1 is assumed by MLAB, and if N and C are both omitted, a value of 2000 is assumed for N by MLAB. If the file contains fewer than N values, MLAB returns a matrix of the minimum size necessary to contain all of the numbers in the file. The read operator does not set a pointer to the

last location read. Thus, even if only part of the file is read, a subsequent READ operation will read from the beginning of the file. Thus, the statement M=READ("PRD1"), reads the whole file named PRD1 (or 2000 values if the file is longer than 2000 numbers long) into a single column matrix M.

```
*M=READ("PRD1")
*TYPE M

MATRIX M:

 6.000000      *
 6.777000      *
 2.200000      *
 .5400000@-5   *
 .5430000@11   *
 1.000000      *
 3.000000      *
 6.000000      *
 6.000000      *
 9.000000      *
 999.0000      *
 3.000000      *
```

The statement N=READ("PRD1",5) reads the first 5 numbers in PRD1 into a single column matrix N.

```
*N=READ("PRD1",5)
*TYPE N

MATRIX N:

 6.000000      *
 6.777000      *
 2.200000      *
 .5400000@-5   *
 .5430000@11   *
```

The statement O=READ("PRD1",12,3) reads the first 12 numbers in PRD1 into a 3 column matrix O.

```
*O=READ("PRD1",12,3)
*TYPE O

MATRIX O:

 6.000000      6.777000      2.200000      *
 .5400000@-5   .5430000@11   1.000000      *
 3.000000      6.000000      6.000000      *
 9.000000      999.0000      3.000000      *
```

EXAMPLE MLAB 6.B - THE PRINT STATEMENT

The MLAB PRINT statement may be used to output the values of MLAB entities to a disk file. This may be done simply to obtain line printer output of some or all of the MLAB entities produced by a session or to create a disk data set which may be read by other MLAB programs or by FORTRAN programs. The statement PRINT O IN PRD2 produces a file named PRD2.LST which contains the value of the MLAB MATRIX O.

```
*PRINT O IN PRD2
```

The following is a listing of the file PRD2.LST, produced by the PRINT statement.

```
MATRIX O:

 6.000000      6.777000      2.200000      *
 .5400000@-5   .5430000@11   1.000000      *
 3.000000      6.000000      6.000000      *
 9.000000      999.0000      3.000000      *
```

This file may be read using the MLAB READ operator, following the same rules as were illustrated in the examples above. Note that MLAB ignores the blanks lines and the line containing the description "MATRIX O". This file could also be read using a FORTRAN program with appropriate format statements. It could be put into a form readable by a FORTRAN namelist read writing it out as a single column matrix and simply replacing the *'s with commas and the @'s with E's.

```
*P=READ("PRD2.LST")
*TYPE P

MATRIX P:

 6.000000      *
 6.777000      *
 2.200000      *
 .5400000@-5   *
 .5430000@11   *
 1.000000      *
 3.000000      *
 6.000000      *
 6.000000      *
 9.000000      *
 999.0000      *
 3.000000      *
```

APPENDIX A: BUILT-IN OPERATORS AND FUNCTIONS

This appendix contains most of the built-in operators and functions of MLAB as they are listed in the MLAB Manual written by Gary Knott of NIH. This appendix is, in fact, essentially a transcription of the listing given by Knott.

A list of the operators follows:

OPERATOR	DESCRIPTION
A + B	SUM OF A AND B. IF A IS A MATRIX AND B IS A SCALAR, B WILL BE ADDED TO EVERY ELEMENT IN A. IF A AND B ARE MATRICES, ROWS AND/OR COLUMNS OF ZEROES WILL BE ADDED IF NECESSARY TO MAKE A AND B HAVE THE SAME DIMENSIONS.
A - B	DIFFERENCE OF A AND B. MATRICES ARE HANDLED AS WITH +.
-A	EQUAL TO 0 - A.
A * B	PRODUCT OF A AND B. MATRICES WILL BE MADE CONFORMAL BY ADDING ROWS OR COLUMNS OF ZEROES IF NECESSARY. SCALARS TIMES MATRICES ARE LEGAL.
A / B	QUOTIENT OF A AND B. B MUST NOT BE A MATRIX.
A ^ B	A TO THE POWER B. B MUST NOT BE A MATRIX. IF A IS A MATRIX, B MUST BE AN INTEGER. IF B<0 THEN THE APPROPRIATE POWER OF THE MOORE-PENROSE GENERALIZED INVERSE OF A EXPANDED FOR CONFORMALITY AS NEEDED, WILL BE RETURNED. (A CIRCUMFLEX IS OFTEN SUBSTITUTED FOR "^".)
ABS A	ABSOLUTE VALUE OF A IF A IS A SCALAR. IF A IS A MATRIX THE LENGTHS OF THE ROWS OF A ARE RETURNED IN A ONE COLUMN MATRIX. BY LENGTH, WE MEAN THE ORDINARY EUCLIDEAN LENGTH GIVEN BY THE SQUARE ROOT OF THE SUM OF THE SQUARED COMPONENTS.
A < B	1 IF A<B ELSE 0. IF A OR B IS A MATRIX, THE RESULT IS A MATRIX M WITH M(I,J) - A(I,J)<B(I,J)
A > B	1 IF A>B ELSE 0. MATRICES ARE HANDLED AS ABOVE.
A = B	1 IF A=B ELSE 0. MATRICES ARE HANDLED AS ABOVE.
A <= B	1 IF A <=B ELSE 0. MATRICES ARE HANDLED AS ABOVE.
A >= B	1 IF A>=B ELSE 0. MATRICES ARE HANDLED AS ABOVE.
A NOT = B	0 IF A=B ELSE 1. MATRICES ARE HANDLED AS ABOVE.
A AND B	1 IF A>0 AND B>0 ELSE 0. MATRICES ARE HANDLED AS ABOVE.

A OR B — 1 IF A>0 OR B>0 ELSE 0. MATRICES ARE HANDLED AS ABOVE.

IF A THEN B ELSE C
IF A NOT= 0 THEN B ELSE C.
WHEN AN IF-THEN-ELSE EXPRESSION IS EVALUATED, ONLY THE CORRECT EXPRESSION (EITHER THE THEN-EXPRESSION OR THE ELSE-EXPRESSION) WILL BE EVALUATED.

A:B — A AND B ARE COERCED TO SCALARS. THE RESULT IS A ONE COLUMN MATRIX WITH THE FIRST ELEMENT EQUAL TO A, THE SECOND ELEMENT EQUAL TO A+1, ETC. THE LAST ELEMENT IS <= B.

A:B:C — A, B, AND C ARE COERCED TO SCALARS. THE RESULT IS A ONE COLUMN MATRIX WITH THE FIRST ELEMENT EQUAL TO A, THE SECOND ELEMENT = A+C, ETC. THE LAST ELEMENT IS <=B IF B IS POSITIVE AND >=B IF B IS NEGATIVE.

The following operators apply only to matrices:

OPERATOR — DESCRIPTION

A(I,J) — MATRIX-SUBSCRIPTING USES THE TRADITIONAL FUNCTIONAL NOTATION. I AND J ARE SCALAR EXPRESSIONS AND A(I,J) IS THE SCALAR VALUE IN THE I-TH ROW AND J-TH COLUMN OF A. IF (I,J) INDEXES A NON-EXISTENT VALUE, AN ERROR MESSAGE RESULTS. A(I) IS DEFINED TO MEAN A(I,1).

A′ — TRANSPOSE OF A. FOR ANY I,J, A′(I,J) = A(J,I).

A ROW B — IF B IS A SCALAR, A ROW B IS THE B-TH ROW OF A. IF B IS A MATRIX, A ROW B IS THE MATRIX FORMED BY TAKING FROM A THE ROWS SPECIFIED BY THE ELEMENTS OF B. B MAY BE SPECIFIED AS A PARENTHESIZED SEQUENCE OF SCALAR AND/OR MATRIX VALUES WHICH IS TAKEN AS AN IMPLICIT ARGUMENT TO THE BUILT-IN OPERATOR LIST DESCRIBED BELOW.

A COL B — SIMILAR TO A ROW B. COLUMNS OF A RATHER THAN ROWS ARE TAKEN. NOTE A ROW I COL J IS A 1 BY 1 MATRIX, WHEREAS A(I,J) IS A SCALAR.

A & B — THE ROW CONCATENATION OF A AND B.

```
IF A =  1  2     AND B =   5  6
        3  4               7  8
THEN A & B =   1   2
               3   4
               5   6
               7   8
```

IF B IS A SCALAR, A CONFORMAL MATRIX WHOSE ELEMENTS ALL HAVE THE VALUE B IS USED IN PLACE OF THE SCALAR. IF A IS A SCALAR A SIMILAR DEFINITION APPLIES.

A &´ B	THE COLUMN CONCATENATION OF A AND B. IF A AND B ARE AS IN THE DESCRIPTION ABOVE, THEN A &´ B= 1 2 5 6 3 4 7 8 IF A OR B ARE SCALARS THE REMARKS CONCERNING & APPLY.
F ON M	F ON M IS THE 1 COLUMN MATRIX OF THE FUNCTION F APPLIED SUCCESSIVELY TO EACH ROW OF THE MATRIX M TAKEN AS ARGUMENTS.
A ^^ B	B IS COERCED TO A SCALAR, A IS COERCED TO A MATRIX, AND THE RESULT IS A & A & ... & A WHERE A OCCURS B TIMES. THUS 1 ^^ 5 = [1,1,1,1,1]´. (A CIRCUMFLEX IS OFTEN SUBSTITUTED FOR "^" ON TERMINAL KEYBOARDS.)
A # B	A AND B MUST BE 3 COLUMN MATRICES. THEN A 3 COLUMN MATRIX WITH MIN(NROWS(A),NROWS(B)) ROWS IS RETURNED WHOSE I-TH ROW IS THE VECTOR CROSS PRODUCT OF THE I-TH ROW OF A AND THE L-TH ROW OF B.
A *´ B	THE RESULT IS A MATRIX WITH MIN(NROWS(A),NROWS(B)) ROWS AND MAX(NCOLS(A),NCOLS(B)) COLUMNS. THE (I,J) ELEMENT OF THE RESULT IS A(I,((J-L) MOD NCOLS(A))+1)* B(I,((J-L) MOD NCOLS(B))+1). THIS IS A USEFUL OPERATION FOR MULTIPLYING EACH ROW OF A MATRIX A BY A SCALAR WHICH DEPENDS ON THAT ROW.

Unless otherwise noted, the functions below require scalar arguments. If matrices are given, no error will be detected, and moreover, nonsense will result!

FUNCTION	DESCRIPTION
SIN(X)	SINE OF X. X IS IN RADIANS.
SIND(X)	SINE OF X. X IS IN DEGREES.
COS(X)	COSINE OF X. X IS IN RADIANS.
COSD(X)	COSINE OF X. X IS IN DEGREES.
ATAN(X)	ARC TANGENT OF X. (IN -PI/2 TO PI/2).
ATAN2(Y,X)	ARC TANGENT OF Y/X. COMPLETE RESULT IN -PI TO PI. ATAN2(Y,X) IS THE ANGLE FORMED BY THE VECTOR TO THE POINT (X,Y) AND THE X-AXIS.
ASIN(X)	ARC SINE OF X. (IN -PI/2 TO PI/2).
ACOS(X)	ARC COSINE OF X.
SINH(X)	HYPERBOLIC SINE OF X.
COSH(X)	HYPERBOLIC TANGENT OF X.
SQRT(X)	SQUARE ROOT OF X.
LOG(X)	NATURAL LOG OF X.

LOG10(X) COMMON LOG OF X.

EXP(X) E TO THE POWER X.

INT(X) THE GREATEST INTEGER LESS THAN OR EQUAL TO X.

MOD(X,Y) INT(X)-INT(INT(X)/INT(Y))*INT(Y)

RAN(X) A RANDOM NUMBER STRICTLY BETWEEN 0 AND 1 IS RETURNED. IF X = 0 THE NEXT NUMBER IN THE CURRENT SEQUENCE WILL BE RETURNED. IF X IS NOT 0, A NEW SEQUENCE WILL BE STARTED. A COMPUTATION INVOLVING X WILL BE USED TO SET THE FIRST VALUE.

LOGGAMMA(X) THE LOG OF THE GAMMA FUNCTION ON X IS RETURNED, WHERE X MUST BE POSITIVE. FOR COMPUTING THE GAMMA FUNCTION WITH X NEGATIVE.

TYPEOUT(X) THE VALUE OF X IS RETURNED, BUT AS A SIDE-EFFECT. IT IS ALSO TYPED OUT! THIS IS A USEFUL FUNCTION FOR DEBUGGING OTHER FUNCTIONS, WHEN SEARCHING FOR THE CAUSE OF OVERFLOWS, FOR EXAMPLE.

INTEGRAL(X,A,B,E)
THE DEFINITE INTEGRAL OF THE EXPRESSION E WITH RESPECT TO X ON THE INTERVAL [A,B] IS RETURNED. INTEGRAL CAN ONLY BE USED IN FUNCTION BODIES.

SUM(I,A,B,E)
THE SUM OF THE EXPRESSION E FOR I RUNNING FROM A TO B IS RETURNED. SUM CAN ONLY BE USED IN FUNCTION BODIES.

ROOT(X,A,B,E)
E IS AN EXPRESSION INVOLVING X. A ROOT OF E(X) IS RETURNED. MORE PRECISELY, A SCALAR VALUE, V, IS RETURNED SUCH THAT E(X) IS 0 WITH X=V. V LIES IN THE INTERVAL [A,B]. IF NO ROOT IS FOUND, A LARGE NEGATIVE VALUE IS RETURNED (-.171...@39) AND A WARNING IS TYPED OUT. ROOT CAN ONLY BE USED IN FUNCTION BODIES. FOR EXAMPLE, TYPING [FUNCTION F(A)=ABS ROOT (X,0,A,X*X-A)] DEFINES F(A) AS THE SQUARE ROOT OF A FOR POSITIVE A.

OPERATOR DESCRIPTION

EVAL(X,A,E) A AND E ARE SCALAR EXPRESSIONS AND X IS A VARIABLE NAME. THE RESULTING VALUE IS THE VALUE OF THE EXPRESSION E WITH THE EXPRESSION A REPLACING EACH OCCURRENCE OF THE SYMBOL X IN E. THUS EVAL IS A SUBSTITUTION OPERATOR. IT IS USEFUL IN COMPACTLY DEFINING FUNCTIONS, AND ARISES AUTOMATICALLY WHEN THE ROOT OPERATOR IS DIFFERENTIATED. EVAL CAN ONLY BE USED IN FUNCTION BODIES.

LOOKUP(A,M) M IS A TWO COLUMN MATRIX IN SORT BY COLUMN 1. A IS A SCALAR VALUE. THE VALUE CORRESPONDING TO A IN THE CURVE GIVEN BY M IS RETURNED.
LOOKUP(A,M)=INTERPOLATE(M,A)[1,2], EXCEPT THAT SIMPLE LINEAR INTERPOLATION IS USED.
LOOKUP INVOKED WITH ITS ARGUMENTS REVERSED MAY CAUSE MLAB TO CRASH!

NROWS(M) THE NUMBER OF ROWS IN THE MATRIX M IS RETURNED.
THIS IS USEFUL IN DEFINING GENERAL SUMS

NCOLS(M) THE NUMBER OF COLUMNS IN THE MATRIX IS RETURNED.

LIST(R1,R2,...)
LIST MAY TAKE ANY NUMBER OF ARGUMENTS. IT RETURNS A ONE-COLUMN MATRIX. IF R1 IS A SCALAR, THE FIRST ELEMENT OF THE RESULT IS THE VALUE OF R1. IF R1 IS AN M X N MATRIX, THE FIRST M*N ELEMENTS OF THE RESULT ARE THE ELEMENTS OF R1, TAKEN BY ROWS. THE VALUE OR VALUES SPECIFIED BY R2 ARE TREATED SIMILARLY AND CONCATENATED ON THE RESULT, AND SO ON. IN PARTICULAR LIST(M) MAKES A VECTOR LIST OF THE ELEMENTS OF THE MATRIX M, TAKEN ROWWISE.

POINTS(F,M) F IS A FUNCTION, M IS A MATRIX. THE RESULT OF POINTS IS A MATRIX HAVING ITS FIRST N COLUMNS EQUAL TO THE N COLUMNS OF M. COLUMN N+1 IS THE RESULT OF F EVALUATED SUCCESSIVELY ON THE ROWS OF M. POINTS(F,M)=M&'(F ON M).
POINTS IS VERY USEFUL IN OBTAINING MATRICES FROM FUNCTIONS FOR DRAWING CURVES.

READ("DEV:FILE.EXT[P,P]",N,C)
THE FILE NAMED "FILE.EXT[P,P]" WILL BE OPENED ON DEVICE "DEV" AND NUMBERS WILL BE READ FROM THE FILE UNTIL END OF FILE IS REACHED OR N*C NUMBERS HAVE BEEN READ.
NUMBERS MAY BE SEPARATED BY BLANKS OR COMMAS OR ALMOST ANYTHING ELSE! THE "@" NOTATION IS USED FOR POWER-OF-TEN SCALING; THE FORTRAN "E" NOTATION WILL NOT WORK AS INTENDED!
A C COLUMN MATRIX OF THE NUMBERS READ IS RETURNED. THE NUMBERS READ ARE USED TO SUCCESSIVELY FORM THE ROWS OF THE RESULT MATRIX. IF THE NUMBER OF VALUES READ IS NOT A MULTIPLE OF C, THE FINAL ROW OF THE RESULT HAS ITS LAST UNFILLED SLOTS SET TO ZERO. THE RESULT MATRIX WILL HAVE AT MOST N ROWS, AND MAY HAVE FEWER, IF LESS THAN N*C NUMBERS ARE READ. THE ARGUMENT C MAY BE OMITTED, WHEREUPON C=1 IS ASSUMED, AND WHEN C IS OMITTED, THE ARGUMENT N MAY ALSO BE OMITTED, WHEREUPON N=2000 IS ASSUMED.
READ("TTY:") WILL CAUSE NUMBERS TO BE READ FROM THE TELETYPE TO FORM THE RESULT. THE LAST NUMBER MUST BE FOLLOWED BY <CONTROL>Z.
IF DEV IS NOT GIVEN, "DSK" IS ASSUMED. IF [P,P] IS NOT GIVEN, THE CONTROLLING USER IS ASSUMED.
READ(A), WHERE A IS NOT QUOTED, WILL WORK. THE FILE NAMED A WITH NO EXTENSION WILL BE ACCESSED.
LIKE ALL OTHER FUNCTIONS DEFINED HERE, THE READ OPERATOR WHEN APPLIED DENOTES AN EXPRESSION, THUS IT IS USUALLY USED AS THE RIGHT-HAND SIDE OF AN ASSIGNMENT STATEMENT WHICH DEFINES A MATRIX AS: E.G., M_READ("F.DAT").

INTERPOLATE(M,N)
THE MATRIX M IS A TWO-COLUMN MATRIX WHOSE ROWS REPRESENT POINTS IN THE PLANE. N IS A COLUMN VECTOR OF X-VALUES. INTERPOLATE WILL PASS A SMOOTH CURVE THROUGH THE POINTS OF M AND THEN RETURN THE MATRIX OF POINTS WHOSE FIRST COLUMN IS N, AND WHOSE SECOND COLUMN IS THE CORRESPONDING Y-VALUES READ FROM THE INTERPOLATING CURVE FOR M. M MUST BE IN SORT ON COLUMN 1, AND IT MUST BE POINTS FROM THE GRAPH OF A FUNCTION I.E., SINGLE-VALUED.
THE INTERPOLATING CURVE IS, IN FACT, A CUBIC SPLINE, E.G., A PIECE-WISE JOINING OF SEVERAL CUBIC POLYNOMIALS.

HISTO(M) A CURVE MATRIX FOR THE GRAPH OF AN APPROPRIATE STEP-FUNCTION APPROXIMATION OF THE CURVE GIVEN BY THE CURVE MATRIX M IS RETURNED. SUCH A STEP-FUNCTION CURVE IS CALLED A HISTOGRAM.
M MUST BE IN SORT ON COLUMN 1.

CURVEMATRIX(C)
A COPY OF THE CURVE MATRIX ASSOCIATED WITH THE CURVE C WILL BE RETURNED.

LABELMATRIX(C)
A COPY OF THE LABEL MATRIX ASSOCIATED WITH THE CURVE C WILL BE RETURNED.

INTEGRATE(F1 DIFF X, F2 DIFF X, ..., FN DIFF X,M)
THE FUNCTIONS F1, F2, ... , FN DEFINED BY THE SYSTEM OF DIFFERENTIAL EQUATIONS SPECIFIED BY THE DERIVATIVES F1 DIFF X, ...,, FN DIFF X ARE INTEGRATED USING THE INITIAL CONDITIONS SPECIFIED BY PREVIOUSLY GIVEN INTITIAL STATEMENTS FOR F1, ..., FN AND THE VALUES OF M, F1, F1 DIFF X, F2, F2 DIFF X, ..., FN, FN DIFF X ON THE VALUES IN THE COLUMN VECTOR M ARE RETURNED AS THE SUCCESSIVE COLUMNS OF THE RESULT MATRIX. ONLY FIRST ORDER SYSTEMS CAN BE SOLVED.
THE METHOD USED AND THE ERROR IN THE SOLUTION CAN BE AUTOMATICALLY OR MANUALLY SET.
WHILE THE DIFFERENTIAL EQUATION SOLVER IS RUNNING, THE LETTER "R" MAY BE TYPED, AND A REPORT OF THE DEQ-SOLVER'S PROGRESS AND STATE WILL APPEAR. THE LETTER "Q" MAY ALSO BY TYPED, IN WHICH CASE, THE DEQ-SOLVER WILL STOP IMMEDIATELY AND ZEROES WILL BE SUPPLIED AS THE VALUES WHICH WERE NOT COMPUTER.

SORT(M,X) THIS FUNCTION RETURNS A MATRIX CONSISTING OF THE ROWS OF M INTERCHANGED SO THAT ELEMENTS OF COLUMN X ARE IN INCREASING ORDER.

SMOOTH(M) M IS A TWO COLUMN MATRIX REPRESENTING A CURVE OF "NOISY" DATA. A 5 POINT VARIABLE-INTERVAL HYPERBOLIC SMOOTHING OPERATION IS APPLIED TO M AND THE RESULTING CURVE MATRIX IS RETURNED AS THE VALUE OF SMOOTH(M).
THE ENDS OF THE SMOOTHED CURVE ARE PROBABLY UNTRUSTWORTHY.

DIAG(M) THE ONE COLUMN MATRIX CONSISTING OF THE LONGEST MAIN DIAGONAL OF M IS RETURNED.

COMPRESS(M) A COPY OF M WITH ALL ROWS, WHOSE FIRST ELEMENT IS 0, REMOVED IS RETURNED.
THIS IS A USEFUL DATA-PROCESSING OPERATOR.

SVD(M) M MUST BE AN A BY B MATRIX WITH A >= B. A MATRIX OF THE COMPONENTS OF THE SINGULAR-VALUE DECOMPOSITION OF M IS RETURNED. THIS IS AN A+2B BY B MATRIX H, SUCH THAT THE MATRIX EQUATION U * P * V' = M HOLDS WHERE:

U = H ROW 1:A,
P = H ROW (A+1):(A+B),
V = H ROW (A+B+1):(A+B+B),

AND U AND V ARE ORTHOGONAL MATRICES AND P IS A DIAGONAL MATRIX CONTAINING THE SINGULAR VALUES OF M. WHEN M IS SYMMETRIC (I.E., M=M'), THE SINGULAR VALUES IN P ARE THE ABSOLUTE VALUES OF THE EIGENVALUES OF M, AND THE COLUMNS OF V ARE THE EIGENVECTORS. IN ANY CASE, THE PRODUCT OF THE SINGULAR VALUES IS THE ABSOLUTE VALUE OF THE DETERMINANT OF M.

CROSS(M,N) AN NROWS(M)*NROWS(N) BY NCOLS(M)*NCOLS(N) MATRIX IS RETURNED. EACH ROW OF M IS COLUMN-CONCATENATED WITH EVERY ROW OF N AND THE RESULTING VECTORS FORM THE ROWS OF CROSS(M,N). THIS IS THE ANALOG OF THE SET-THEORETIC CROSS-PRODUCT.

MESH(M,N) A 2*MAX(NROWS(M), NROWS(N)) BY MAX (NCOLS(M), NCOLS(N)) MATRIX IS RETURNED. THE ROWS OF M AND N ARE TAKEN ALTERNATELY TO FORM THE ROWS OF MESH(M,N).

ROTATE(M,X) THE ROTATED MATRIX, M ROW((NROWS(M)+1-X):NROWS(M), 1:NROWS(M)-X) IS RETURNED.

COV(M) M IS TREATED AS A LIST OF NROWS(M) POINTS WITH NCOLS(M) COMPONENTS. EACH ROW OF M IS THUS AN NCOLS(M) DIMENSIONAL OBSERVATION. THE COVARIANCE MATRIX OF THESE OBSERVATIONS IS THE VALUE COV(M). COV(M) IS AN NCOL (M) BY NCOLS(M) MATRIX AND COV(M)[I,J] IS THE UNBIASED SAMPLE COVARIANCE OF THE COORDINATE OBSER- IN M COL I AND M COL J. NOTE IF M HAS ONLY ONE COLUMN, COV(M) [1,1] IS THE VARIANCE OF THE VALUES IN M.

CORR(M) CORR(M) IS THE CORRELATION MATRIX OF THE OBSERVATIONS WHICH ARE THE ROWS OF M. CORR(M)[I,J]=COV(M)[I,J]/ SQRT(COV[I,I]*COV[J,J]).

DATATYPE(E) THE TYPE OF THE DATA OBJECT E IS RETURNED AS AN INTEGER CODE WITH 1 FOR UNKNOWN, 2 FOR CONSTANT, 4 FOR SCALAR VARIABLE, 5 FOR A FUNCTION, 6 FOR A MATRIX, 7 FOR A WINDOW,, 8 FOR A CURVE, 9 FOR AN INITIAL, 15 FOR TEXT, 16 FOR CONSTRAINTS, 18 FOR A SURFACE, AND 19 FOR A 3D-WINDOW. THIS OPERATOR MAY NOT BE USED IN FUNCTION BODIES CORRECTLY.

The following table gives the LINETYPE and POINTTYPE code for the two-dimensional DRAW command:

POINTTYPE SYMBOL		LINETYPE SYMBOL	
Ø	NULL	Ø	NULL LINE
1	VERTICAL BAR	1	SOLID LINE
2	CROSS(+ SIGN)	2	DASHED LINE
3	TRIANGLE	3	LONG DASHED LINE
4	SQUARE	4	MIXED DASHED LINE
5	HORIZONTAL BAR	5	ALTERNATING SOLID
6	ENLARGED DOT		
7	SMALL DOT		
"X"	THE CHARACTER X		

APPENDIX B: THREE-DIMENSIONAL GRAPHICS

MLAB has the capability to represent a three column matrix as a three-dimensional graph. If the user issues a command DRAW M where M is a three or more column matrix, a curve representing the data in the first three columns will be drawn in the DEFAULT3DWINDOW with X-values taken from column 1, Y-values from column 2, and Z-values from column 3.

EXAMPLE MLAB 1

```
* M COL 1=1:10
* M COL 2=2:20:2
* FCT F(X,Y)=X^2+(Y-2)^2
* M COL 3=F ON M COL (1,2)
* DRAW M
* DISPLAY=T4010
* RELEASE DISPLAY
```

Plotting options in three dimensions are listed in the Seventh Edition (July 1977) of the MLAB Manual prepared by DCRT at NIH on pages 126-9. Some of these will be used in the following example.

EXAMPLE MLAB 2

```
* AXES FLOOR NO BOX SEQUENCE
* GET DISPLAY
* DELETE DEFAULT3DWINDOW
```

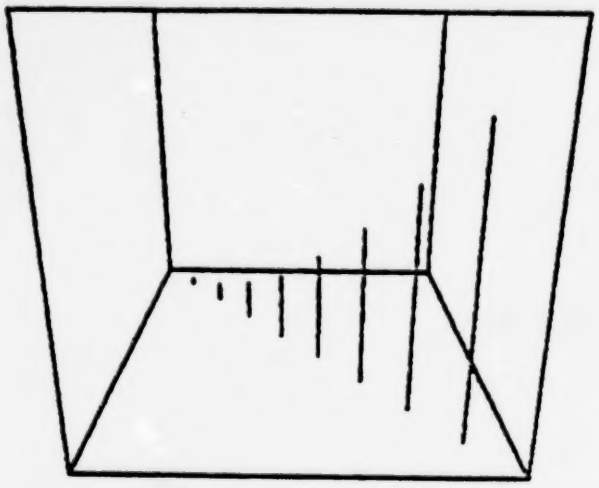
It is important to note that if the two-dimensional DEFAULTWINDOW or any other window exists, they will be superimposed on the DEFAULT3DWINDOW unless they are blanked. The perspective of the graph may be altered by the viewing commands listed on pages 124-5 of the MLAB Manual. The reader is encouraged to try these in order to gain an understanding of their respective functions.

A three-dimensional surface may be displayed by using the following sequence of commands:

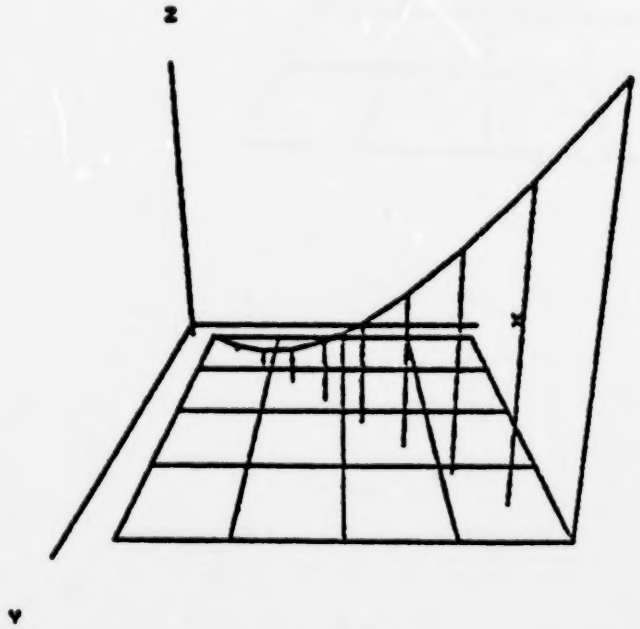
EXAMPLE MLAB 3

```
* XMIN=0; XMAX=10; YMIN=0; YMAX=10
* FCT F(X,Y)=X^2+2*Y^2
* I9=1
* FOR IY=YMIN:YMAX:2 DO(FOR IX=XMIN:XMAX DO(MAT(I9,
  1)=IX;MAT(I9,2)=IY;I9=I9+1); FOR IX=XMAX:XMIN:-1
  DO(MAT(I9,1)=IX;MAT(I9,2)= IF IY=YMAX THEN IY ELS
  E IY+1; I9=I9+1))
* FOR IX=XMIN:XMAX:2 DO(FOR IY=YMAX:YMIN:-1 DO(MAT(
  I9,1)=IX;MAT(I9,2)=IY;I9=I9+1); FOR IY=YMIN:YMAX
  DO(MAT(I9,1)= IF IX=XMAX THEN IX ELSE IX+1;MAT(I9
  ,2=IY;I9=I9+1))
* MAT3D=POINTS(F,MAT)
  DRAW MAT3D
* AXES NO BOX FLOOR NO NEEDLES NET
```

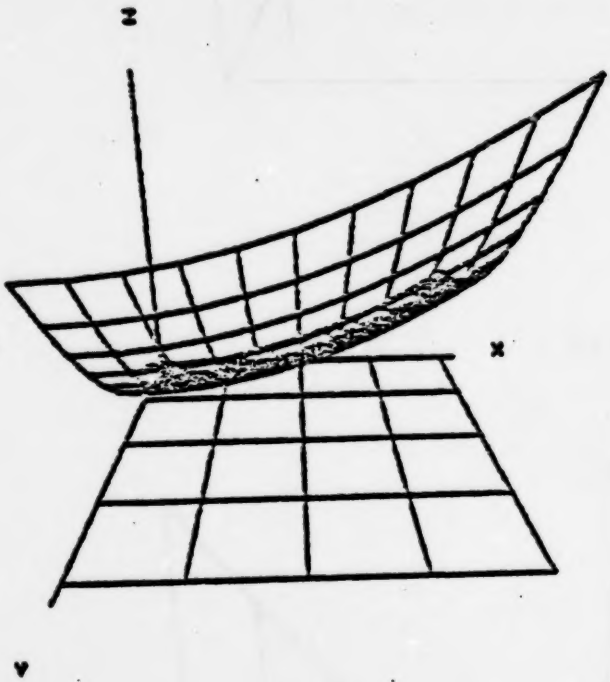
EXAMPLE MLAB 1



EXAMPLE MLAB 2



EXAMPLE MLAB 3



APPENDIX C: USING TELENET

The instructions for logging on found on pages 2-3 are for attaching directly to the FDC system. In certain geographical areas the EPA user should log on through the TELENET system. Please contact your First Data coordinator listed in the Introduction for determining which network is appropriate for your area.

To log on using TELENET:

a. dial the local TELENET number;

b. attach the phone to the acoustic coupler;

c. press two (2) carriage returns;

d. after '@' is typed by the computer, enter your terminal type code followed by a carriage return;

e. type either 'HALF'<cr> or 'FULL'<cr> depending on whether half or full duplex is desired;

f. type 'DISA FOLD'<cr>;

g. type 'C 313 40'<cr>;

h. type another carriage return;

i. follow instructions #4-16 on pages 2-3;

For information on telephone numbers, terminals supported, and terminal type codes, please contact your FDC coordinator.

INDEX

****

NOTE: See APPENDIX for a more complete listing of MLAB commands and their descriptions.

****

account number 2
assign 4
AT 19
BY 22
COLumn 7
concatenation (&) 23,47
constraints 32,37-38,80
CTRL-C 26-27
CTRL-H 26
CTRL-P 2
CTRL-U 26
CTRL-Z 7
curve fitting 1,15-17,31-34,80,84-85
.DAT files 66-68
DEFAULTWINDOW 17,33
DELETE 17,29,69
derivatives 1,80,85
DIFF operator 41-54
differential equations 44-53,79-84
DISPLAY 15
DO 62,72
DRAW 15,46,50,81
error messages 27
escape key 69-71
expression 9
files 1-2,66-72,86-88
FIT 31-34,38,68,80
FOR 62-64
FUNCTION 4
GET DISPLAY 18
IF...THEN...ELSE 55
IN 66
INITIAL 46
input of data 31,69-71,86
INTEGRAL 41-42,44,59
INTEGRATE 46-48,50-54,81,85
INTERPOLATE 75-76
inverse 8,11,13
linear equations 13
LINETYPE 14-15,19,21,54
LIST 7-9,79
logging in 3
logging out 3

loops 62-63
.LST files 88
Marquardt-Levenberg 1,31
matrices 1,8-11,23,35-54,62-63,66,76-81
monitor mode 26,66,68-69
ON operator 33
output of data 88
POINTS 16-17,20,22-23,76
POINTTYPE 15,19,21
PRINT 69,88
PRINT IN 88
QUIET FIT 38,68
READ 7,86-88
RELEASE DISPLAY 17
ROOT 25,59-60
ROW 7,10
SAVE 66
scalars 4,8,10-11,28,41
SIZE 19
solutions (systems of equations) 13-14,41-52,83-84
statements 3,56,72
STRING 19-21
sum of squares 36,65,85
SUM 62
TO 31
TTYDRAW 22-24
TYPE 4,7,47,62,64,66-67
USE 66,68